A TRIDENT SCHOLAR PROJECT REPORT

NO. 222

"An Application of Fuzzy Logic Control to a Classical Military Tracking Problem"

UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

U.S.N.A. - Trident Scholar Project report; no. 222 (1994)

"An Application of Fuzzy Logic Control to a Classical Military Tracking Problem"

by
Midshipman 1/C Erik S. Smith, Class of 1994
U.S. Naval Academy
Annapolis, Maryland

Carl E. Wick

Adviser: Assistant Professor Carl E. Wick
Department of Weapons and Systems Engineering

Accepted for Trident Scholar Committee

Francis D Correll

Chair

19 May 1994

Date

USNA-1531-2

REPORT DOCUMENTATION PAGE

Form Approved
OMB no. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
19 May 1994

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE An application of fuzzy logic control to a classical military tracking problem

5. FUNDING NUMBERS

6. AUTHOR(S)
Smith, Erik S.

7. PERFORMING ORGANIZATIONS NAME(S) AND ADDRESS(ES)
U.S. Naval Academy, Annapolis, MD

8. PERFORMING ORGANIZATION REPORT NUMBER USNA Trident Scholar project report; no. 222 (1994)

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Accepted by the U.S. Trident Scholar Committee

12a. DISTRIBUTION/AVAILABILITY STATEMENT
This document has been approved for public release; its distribution is UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)
Fuzzy logic is an emerging field of theory and application which holds great promise for the control of systems - especially those systems which cannot be described mathematically, or which are nonlinear in nature. Fuzzy logic's strength lies in its heuristic approach to control. Instead of requiring complex mathematical equations which describe a system's behavior, fuzzy logic allows systems designers to use a set of common sense, plain-English rules to invoke a desired system response. The purpose of this project was to explore fuzzy logic as a way to effect control of a target tracking system. The military tracking problem is one that has been well studied, and many solutions using various means of control have been successfully implemented. These control methods, however, are reaching the limits of their application. Fuzzy logic offers an exciting alternative solution to this problem. In pursuit of this project, an optical tracking platform was designed and built. A fuzzy logic control system was also developed and implemented. This system used information about a target laser's position and rate of change of positions with respect to the tracking platform in two dimensions - elevation and azimuth - in order to arrive at its control decisions.

14. SUBJECT TERMS fuzzy logic, fuzzy logic control, tracking problem

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

Abstract

Fuzzy logic is an emerging field of theory and application which holds great promise for the control of systems - especially those systems which cannot be described mathematically, or which are nonlinear in nature. Fuzzy logic's strength lies in its heuristic approach to control. Instead of requiring complex mathematical equations which describe a system's behavior, fuzzy logic allows systems designers to use a set of common sense, plain-English rules to invoke a desired system response.

The purpose of this project was to explore fuzzy logic as a way to effect control of a target tracking system. The military tracking problem is one that has been well studied, and many solutions using various means of control have been successfully implemented. These control methods, however, are reaching the limits of their application. Fuzzy logic offers an exciting alternative solution to this problem.

In pursuit of this project, an optical tracking platform was designed and built. A fuzzy logic control system was also developed and implemented. This system used information about a target laser's position and rate of change of position with respect to the tracking platform in two dimensions - elevation and azimuth - in order to arrive at its control decisions.

Keywords: **Fuzzy Logic, Fuzzy Logic Control, Tracking Problem**

Acknowledgments

I would especially like to thank my Trident Advisor, Assistant Professor Carl Wick, for his support and guidance (and for his unending patience) as I pursued this Trident project. Also, I would like to thank my parents, Gregory and Phoebe Eastman, for their support this past year. Thanks to Ralph Wicklund and Sam Hawkins in Technical Support Division, Carl Owen in the machine shop, and Larry Clemens at the library for the help they have given me - I could not have done this project without them!

A special thanks to William Gotten, with whom I shared a cramped, stuffy, underheated cubicle, and all of the frustrations of two semesters of Trident!

Table of Contents

Section 1 - The Tracking Problem

The tracking problem is one of great importance to the United States Navy. The reason for this is that all weapons systems aboard naval vessels require control algorithms to align them with their intended targets. For instance, a control system is needed to quickly align the Phalanx Close-In-Weapons-System (CIWS) with incoming hostile aircraft and anti-ship missiles. Gunmounts aboard ships also require control in order to track targets in a wide variety of roles, including Naval Gunfire Support (NGFS), Anti-Surface Warfare (ASW), and limited Anti-Air Warfare (AAW).

A tracking problem involves two primary components: a **target**, and a **tracking platform**, as shown in Figure 1. The tracking platform is the component which receives information about the target from the outside world and uses that information to make control decisions about how to align the platform with the target. The target is the component which moves in relation to the tracking platform, and which the tracking platform must follow.

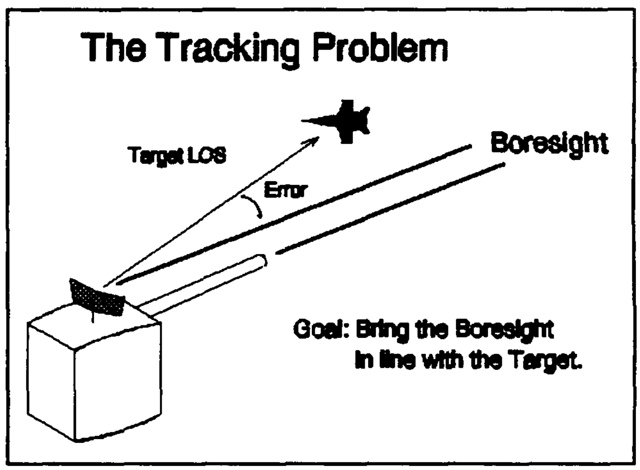


Figure 1

The objective of the tracking problem is to reduce the difference, or the **error**, between the line of sight of a tracking platform (the boresight of the weapon) and the position of an object being tracked (the target line-of-sight, or LOS) in order to align a tracking platform with its target.

In this Trident project, a two-dimensional optical tracking platform was used to obtain azimuth and elevation (x and y) information about a target laser. Two types of data were obtained from the system optics for each of the two axes (azimuth and elevation): position, and rate of change of the position (derivative or velocity) of the laser image with respect to the tracking platform. The specific objective of this application was to center the image of the target laser in the optics of the tracking platform using fuzzy logic as the control for the tracking device.

Section 2 - Fuzzy Logic - An Introduction

Fuzzy logic is a way of mathematically analyzing the uncertainty of information; that is, fuzzy logic is a way of dealing with information that is "gray" in nature. Fuzzy logic excels in dealing with information that cannot be described as being a full member of just one category, but can be described as being a partial member of two or more categories. The method fuzzy logic uses to achieve this result is by breaking information into well-defined categories and by determining the degree of membership of the information within those categories.

Fuzzy logic control extends the principles of fuzzy logic to the solution of a control problem. In addition to assigning information to categories and quantifying the membership of the information within those categories, fuzzy logic control uses a set of linguistic **rules** which incorporate the intuitive knowledge of the system designer about a system's operation. A fuzzy logic system is thus sometimes called an "expert" system because the **rule base** (also called the **Fuzzy Associative Matrix**, or **FAM**) describes the decisions a human operator would make in the control of a system.

2.1 A Short History of Fuzzy Logic

Fuzzy logic was born in 1965 with the publication of Lofti Zadeh's landmark paper, "Fuzzy Sets".[1] Human beings, Zadeh observed, make hundreds of decisions every day based on limited information. These observations grew into the concept of "fuzzy logic", the term Zadeh coined to describe a method which models the way human beings analyze and employ information that is "fuzzy" or ambiguous in nature. "Fuzzy logic control" was a phrase later developed which describes the extension of fuzzy logic to the solution of a system control problem.

For about twenty years after Zadeh's initial work on the subject was published, fuzzy logic remained relatively unknown. Even though fuzzy logic had potential for application in many problems, scientists and engineers in the United States distrusted its use. The word "fuzzy" created an image in their minds of a concept that seemed too imprecise to be of any practical use.[2]

Serious interest in fuzzy logic did not develop until the mid nineteen-eighties, when Japanese engineers successfully applied fuzzy logic to a wide range of control problems, including high-speed train braking and automatic camera focusing.[3] Fuzzy logic did not make inroads in the United States until quite recently - the past four or five years - after the Japanese had already proven the advantages of fuzzy logic systems.[4]

Today, fuzzy logic finds application in problems which can be divided into two broad categories: pattern recognition problems (such as handwritten character recognition) and classical control applications (such as high-speed train braking and automatic camera focusing).[5] This Trident project focuses on fuzzy logic in the latter, more traditional control sense, to a military tracking problem.

2.2 Why Fuzzy Logic For This Project?

There is one important motivation for using fuzzy logic as the control algorithm for this research project. Pacini and Kosko have described an application of fuzzy logic to a two-dimensional tracking problem. However, much of their work on this problem to date has been theoretical, using computer generated models in well-defined, carefully controlled simulations.[6] This Trident project was a perfect opportunity to take this problem and apply it to a physical system in a real-world setting.

2.3 Fuzzy logic - What is it?

Crisp is a term used to describe a value that is definite, as compared to **fuzzy**, which is a term used to describe a value that is ambiguous in nature.

Fuzzy logic is a way of describing the world around us in shades of "gray". This is in contrast to Boolean logic, which is only capable of viewing the world in **crisp** terms of absolute black and absolute white, having no allowance for transition between these two extremes. In fact, it can be proven that Boolean logic is a special case of fuzzy logic, with fuzzy logic being the more general form of logic.[7]

Despite its name, fuzzy logic is neither "fuzzy" nor imprecise in any way. Although fuzzy logic excels in dealing with ambiguous ("gray") information, it does so in a precise manner - by quantifying the degree of ambiguity ("the shade of gray") of that information. The only imprecision with fuzzy logic arises from the way in which fuzzy logic is applied; if the frame of reference does not describe the problem accurately, then the findings of the fuzzy logic system will also be inaccurate.

Because decisions must be made about what rules govern a system of fuzzy logic, fuzzy logic is often described as "heuristic". That is, through observation, a best "guess" must be made as to what rules govern the operation of a system. Fuzzy systems are thus also sometimes termed "expert" systems, because fuzzy systems mimic the decisions human operators would make in the control of those system. Only through repeated observation, analysis, and tuning of a system's rule base can a fuzzy logic system achieve its intended objective.

2.4 Probability and Fuzzy Logic

It was stated above that fuzzy logic deals with uncertainty. While this is true, fuzzy logic is not the same as probability. Probability and fuzzy logic are both terms used to describe uncertainty, but the manner in which each of these concepts deals with uncertainty is radically different. Probability measures the uncertainty present in the occurrence of an event, while fuzzy logic measures the uncertainty in the characteristics of an event that has occurred.[8]

> Fuzzy logic focuses on the ***characteristics*** of an event, while probability focuses on the ***likelihood*** of an event.

> Fuzziness describes event *ambiguity*. It measures the degree to which an event occurs, not whether it occurs. Randomness describes the uncertainty of *event occurrence*. An event occurs or not, and you can bet on it.[9]

An example of probability is "There is a 70% chance of precipitation on Tuesday." This is a statement of prediction - seven times out of ten it is expected to precipitate on Tuesday. An example of fuzzy logic is "The soil is 90% saturated with water." This is a statement of fact - the soil is mostly, but not completely (90%), saturated with water.

2.5 The Fuzzy Logic Control Algorithm

This paper has already examined fuzzy logic, including some of its most fundamental characteristics - but how does it work, and how can it be applied to the control of a system?

A fuzzy logic control algorithm can be divided into three distinct steps: fuzzification, rule evaluation, and defuzzification.

In the first step (referred to as fuzzification), fuzzy **input membership functions** break system input information into categories and assign membership values to those categories. The second step, rule evaluation, contains the **Fuzzy Associative Matrix** (**FAM**), which is a set of rules which describe the desired system operation. Defuzzification is achieved with **output** and **weighting** functions, which bring together all of the information derived from the previous two steps and combine it to obtain a single, crisp control output.

2.51 Fuzzy Input Membership Functions

Consider a world in which all people are described as being either short or tall. A representation of the categories of short and tall might be depicted as shown in Figure 2.

This Boolean representation of height by category works well if a person measures 3'6" in height, because most people would agree that such a person completely fits into the category of SHORT and is completely outside the category of TALL. This binary representation also works well if a person is 6'6" tall. Again, most people would agree that such a person fits completely into the category of TALL and is completely outside the category of SHORT.

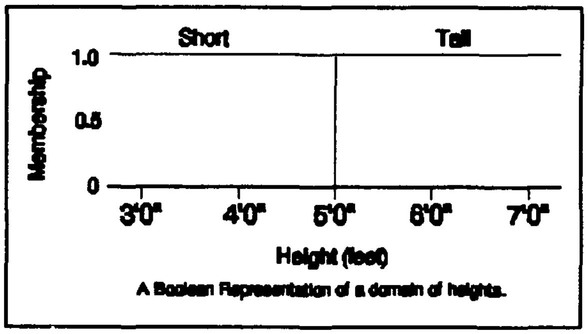


Figure 2

However, a problem occurs with this representation when one tries to fit an individual who is 5'0" in stature into a specific, well-defined category. At this point, if person's height changed by a very small amount, his (or her) category would change abruptly from one case to the other. This is a problem because it is difficult to justify why this particular point (5'0") is the only valid transition point.

A possible solution to this problem is to add a third category so that the representation of height by category is portrayed in Figure 3.

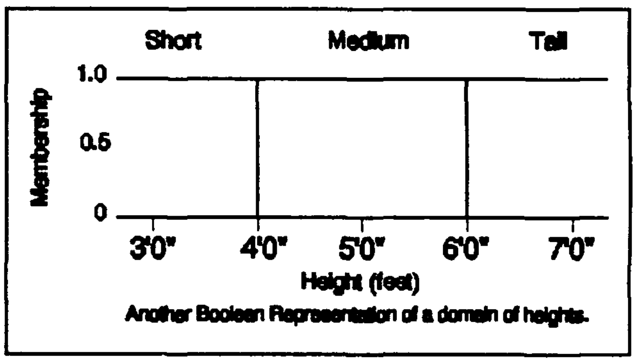
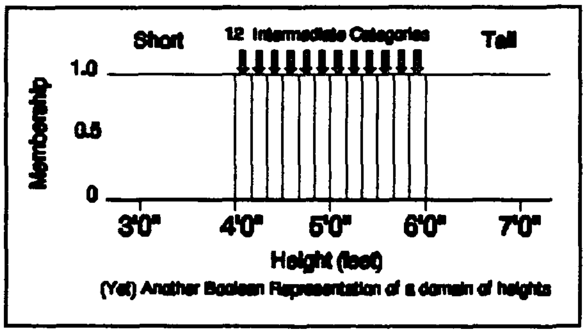


Figure 3

However, the same difficulty arises with this representation. Abrupt transitions still occur between the categories of SHORT and MEDIUM, and between the categories of MEDIUM and TALL.

A final solution to this problem using traditional Boolean techniques would be to further subdivide the domain of heights into smaller and smaller increments, as illustrated in Figure 4.



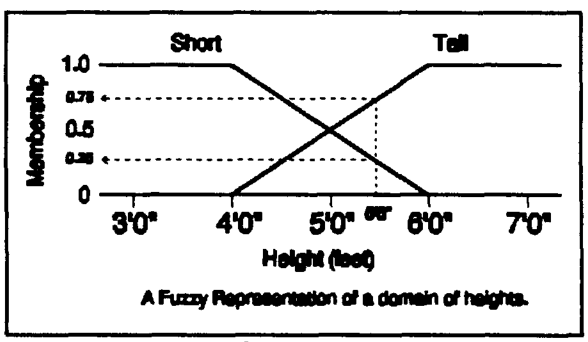
(Yet) Another Boolean Representation of a domain of heights

Figure 4

Eventually, however, increasing the number of categories makes the problem more, instead of less, complex, and a satisfactory solution to the problem is never reached. The reason for this is that the transition between SHORT and TALL blurs as one struggle to conceptualize what each intermediate increment of height signifies.

A fuzzy logic solution to this problem is illustrated in Figure 5. The range of data values (3'0" to 7'0") is called the **input membership function domain**, and the individual categories themselves (**SHORT** and **TALL**) are called **input membership functions**.

Notice in Figure 5 that a smooth transition occurs between the categories of SHORT and TALL, and meaningful information is obtained about a person if his or her height falls within this transition area. A person who is 5'6" tall, for example, will now acquire a value (.25) that indicates the degree of



A Fuzzy Representation of a domain of heights.

Figure 5

membership in the category of SHORT, and another value (.75) indicating how well this height fits in the category of TALL. That is, the statement, "This person is SHORT." would be 25 percent true, and the statement "This person is TALL." would be 75 percent true.

Reflecting back, one realizes that fuzzy logic has accomplished two things: first, it has categorized 5'6" tall into SHORT and TALL. Second, it has assigned **membership** values to those categories, values which fall between 0.0 (completely false) and 1.0 (completely true).

Determining or finding input membership functions is the first step of the fuzzy logic control process - in which a fuzzy algorithm categorizes the information entering a system, and assigns values which represent the degree of membership in those categories. Input membership functions themselves can take any form the designer of the system desires - triangles, trapezoids, bell curves, or any other shape - as long as those shapes accurately represent the distribution of information within the system, and as long as there is a region of transition between adjacent membership functions.

In the tracking problem studied in this Trident project, two variables were considered for each axis - position and derivative of the position of the target laser's image relative to the tracking platform. Each variable was separated into seven input membership functions which described the input domain - NL, NM, NS, ZE, PS, PM, and PL, where N means negative, P positive, L large, M medium, S small, and ZE zero. A representation of the position input membership function domain for the tracking system is shown

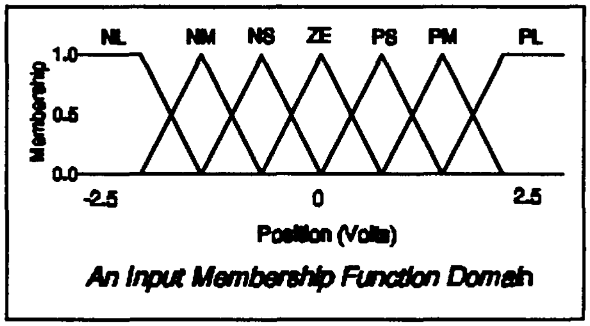


Figure 6

Figure 6. (Units of volts are used because the system photoelectric sensor provided electrical information to describe the position of the laser image.) A representation of the derivative input membership function domain was not provided because it is similar to the representation shown in Figure 6.

2.52 The Fuzzy Rule Base

The second step in the development of a fuzzy logic control system is the determination of the fuzzy rule base, or Fuzzy Associative Matrix. Within the fuzzy rule base lies the soul of a fuzzy control system, for here one can find the heuristic rules which incorporate human knowledge, intuition, and expertise into the control of a system.

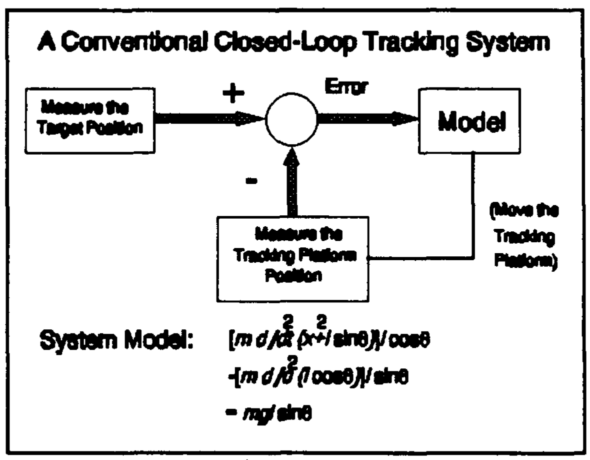


Figure 7

In a conventional control system (illustrated in Figure 7), mathematical equations describe how the system will perform. However, there is a problem with this approach. A particular system may not easily be described mathematically. For example, the system may be nonlinear in nature. Or, even if the system can be modeled accurately, subtle changes in the physical parameters of the system (such as inertia or damping) may substantially change system performance. Another disadvantage to conventional control systems is that they require controllers which contain a great deal of memory and computing power in order to properly implement the mathematical control equations.[10]

In fuzzy logic control (illustrated in Figure 8), the processes which occur in a system must still be well understood. However, fuzzy logic controllers avoid the difficulties conventional controllers encounter because fuzzy logic simplifies the approach to the solution of control problems. This is because a fuzzy system does not require a mathematical model of a system's behavior. Instead, a human operator's expertise is needed in the form of a base of decision-making rules.[11]

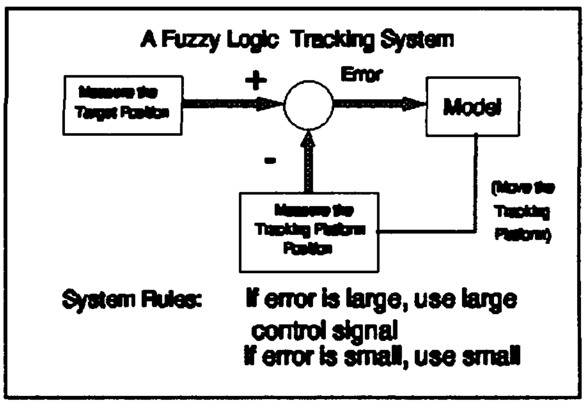


Figure 8

It is useful at this point to explain the similarities and differences between fuzzy logic control and artificial intelligence. Both artificial intelligence and fuzzy logic control use a set of **IF-THEN** rules which describe what action is to be taken if a certain set of conditions is met. Artificial intelligence rule bases, however, have a finite number of control points - one control point for every IF-THEN rule. In a fuzzy rule base, there are still a limited number of IF-THEN rules, but an infinite number of control points is possible because a fuzzy rule base maps membership values to corresponding control values. This means that a fuzzy rule base recognizes information that is fuzzy or partially true in nature, and can partially "fire" or invoke more than one rule at any one time.[12]

It is useful to demonstrate these concepts through an example. In the tracking problem being considered, it is desired to position the tracking platform so that it is in line with its intended target. If the tracking platform is far out of position with respect to its target, then one could make the rule:

If the error is LARGE, then the control output is LARGE.

This makes sense. If the platform is seriously out of line with the target, then a large control force is needed to move the platform quickly back into position. Likewise, if only a small discrepancy exists between the platform and its target, then one could derive the control rule:

If the error is SMALL, then the control output is SMALL.

This, too, makes sense. If there is only a small inconsistency between the control platform and the target, then only a small correction is needed.

Adding directional information, one gives the control outputs further meaning. For instance:

If the error is LARGE POSITIVE, then the control output is LARGE NEGATIVE.

If the error is SMALL NEGATIVE, then the control output is SMALL POSITIVE.

These rules simply mean that if the tracking platform is displaced to one side of the center point, then a force is needed in the opposite direction to bring the platform back in line.

All of the rules above are valid, but they only incorporate knowledge of one input variable, position, in the control decision. The tracking problem considered for this Trident project, however, includes information about two variables - position and rate of change of position. An example of a rule that takes both variables into account is:

IF the error is LARGE POSITIVE *AND* if the rate of change of the error is LARGE NEGATIVE, then the control output is ZERO.

If the target is well to one side of the center point of the tracking device, and if the tracking device is already moving quickly in toward the center point, then little, if any, extra effort is needed by the controller to place the tracking platform back on mark.

For a rule base to be valid, it must incorporate information about every possible condition that the system can be expected to encounter. Each unique combination of conditions will correspond to a control decision in the form of a rule. In this tracking problem, two variables are considered for each axis with each variable breaking its domain into seven input membership functions, or conditions. Thus, there are 49 (7*7) unique combinations of conditions with each combination corresponding to a rule which describes the operation of the fuzzy controller. As illustrated in Figure 9, mapping all of the possible condition combinations in a rule base takes the form of a rule matrix.

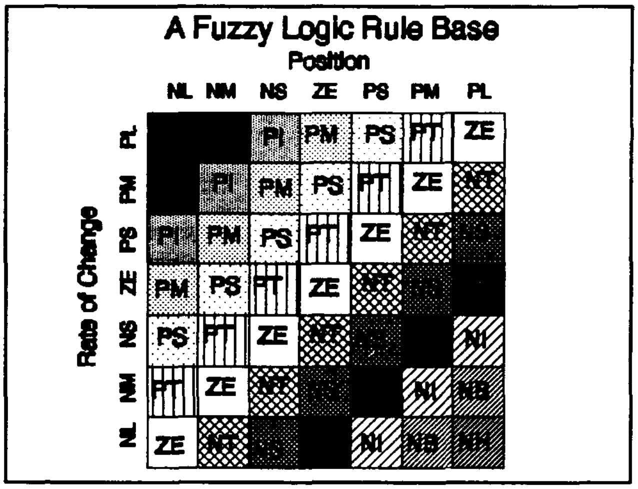


Figure 9

2.53 Fuzzy Minimums and Maximums

This paper has already discussed input membership functions, where information entering a system is categorized and the categories are assigned membership values. This paper has also examined the rule base, the place where decisions are made about how to use the information derived from the input membership functions. However, the question remains - how do these two steps work together?

Consider an example using this project's tracking system. A hypothetical set of one axis' (either elevation's or azimuth's) position and derivative data is contained in

The error is POSITIVE SMALL.	**(.25)**
The error is ZERO.	**(.75)**
The derivative of the error is NEGATIVE SMALL.	**(.40)**
The derivative of the error is ZERO.	**(.60)**

Example 1

Example 1. This information has been "fuzzified" - categorized and assigned membership values - after being obtained by the system optics. This explains why there are two sets of data for both the error (position) and for the derivative. The uppercase letters in Example 1 denote categories of information and the numbers in parentheses denote membership values for those categories.

The four statements found in Example 1 will invoke or "fire" the rules found in Example 2. In Example 2, the numbers in parentheses denote the degree of membership of an input in a particular input membership function. The numbers

If the error is ZERO (.75) *AND* if the derivative of the error is NEGATIVE SMALL (.40), then the control output is POSITIVE SMALL [.40].

If the error is ZERO (.75) *AND* if the derivative of the error is ZERO (.60), then the control output is ZERO [.60].

If the error is POSITIVE SMALL (.25) *AND* if the derivative of the error is NEGATIVE SMALL (.40), then the control output is ZERO [.25].

If the error is POSITIVE SMALL (.25) *AND* if the derivative of the error is ZERO (.40), then the control output is NEGATIVE SMALL [.25].

Example 2

in brackets denote the degree to which a particular rule is invoked. In every case, the degree to which a rule is fired is the ***minimum*** of the membership values of the individual conditions which invoke that rule. Thus, each one of the rules found in Example 2 is fired only to the least degree of its invoking conditions' memberships.

The reason why the minimum was taken of the input conditions to each rule is a postulate of fuzzy logic; in fuzzy logic, the ***AND*** function is the same as taking the minimum of the values of the conditions for the function.[13] For a statement "A ***AND*** B", the fuzzy logic ***AND*** function is a test to determine the extent of membership both input conditions, A and B, share in a fuzzy set. Since the greatest extent that both of these conditions exist in a fuzzy set is the ***minimum*** of the input conditions, the minimum of the conditions is a ken to satisfy a fuzzy ***AND*** function.

As it will become important shortly, the fuzzy ***OR*** function takes the ***maximum*** of the values of the conditions for a function.[14] Using similar reasoning as before, in the statement "A ***OR*** B", the fuzzy logic ***OR*** function is a test to determine the extent of membership either one of the input conditions, A or B, has in a fuzzy set. Since the greatest extent that either one of these conditions exist in a fuzzy set is the ***maximum*** of the input conditions, the maximum of the conditions is taken to satisfy a fuzzy ***OR*** function.

2.54 The Final Step - Weighting and Combining Rules

The final step of the fuzzy logic algorithm is weighting and combining the information obtained from the previous two steps in order to obtain a single, crisp control output. There are several methods which can be used to obtain this output, but the simplest - called the **centroid method**[15] - is to sum the multiples of the values of the rules with their weights and to divide this total by the sum of the weights. The centroid equation is:

$$C_o = \frac{\sum_{i=1}^{j} W_i * R_i}{\sum_{i=1}^{j} W_i}$$

(Where C_o is the control output, R_i is a rule, and W_i is a rule weight.)

Before deriving the final control output, however, one final check must be made of the rules that have been invoked. If two or more rules are fired that have the same value, then the rule which is fired to the greatest degree is taken, and the rest of the rules are discarded. From Example 1 before, one notices that the rule **ZERO** has been invoked twice - once to a degree of .60, and once again to a degree of .25. A choice must be made between one rule ***OR*** the other. Since the fuzzy ***OR*** function states that the maximum of the two rules must be taken, the rule that is fired to the degree of .60 is retained, and the rule that is fired to the degree of .25 is discarded. Example 3 contains the three rules that remain and their corresponding weights.

ZERO	**.60**
POSITIVE SMALL	**.40**
NEGATIVE SMALL	**.25**

Example 3

The last step is to weight and combine the rules. Normally a control rule will correspond to a value, which tells the control system how to respond. For this problem, a motor control voltage would probably be the desired control output, so for the control rule **ZERO** one could attach a value of zero volts, and for the control rules **POSITIVE SMALL** and **NEGATIVE SMALL**, one could attach values of plus and minus five volts, respectively. These values are derived heuristically and incorporate one's "best guess" of how the rule base should represent system operation.

Weighting and combining these values together would take the form:

$$\frac{(.4*5)+(.25*-5)+(.6*0)}{.4+.25+.6} = .60 volts$$

Thus, a small positive voltage would need to be applied to the motor in order to center the axis of the tracking platform on the target laser.

Section 3 - Systems Design and Construction

Design and construction of the tracking system involved three major areas - construction of the tracking platform, design and construction of the optics and optical interfacing circuitry, and design and construction of the fuzzy logic control hardware and software. Each area had its own unique problems and considerations.

3.1 Construction of the Tracking Platform

The tracking platform (Figure 10) was built in two steps. In the first step of construction, the base of the platform was built and a Galil motor was mounted to control the azimuth axis of the tracking platform.

The second step of construction of the tracking platform progressed as follows. First, a cradle was made capable of mounting a Celestron C-90 spotting scope. Rails were installed on the cradle to hug the base of the scope and to prevent unwanted side-to-side motion of the scope. Also, a slot wide enough to accommodate a standard camera screw was cut to mount the scope and to allow precise front-to-back balance adjustment of the scope.

Next, the cradle was mounted on a pair of shafts allowing free movement of the cradle in the elevation axis. The shafts were then placed on shaft bearings pressed into sidewalls. These sidewalls were mounted on a secondary base which was free to rotate around the azimuth axis on a "lazy Susan" swivel bearing. This "lazy Susan" was mounted on the base built in the first step of construction. Control of the elevation axis was achieved by connecting the cradle shaft to a Galil motor mounted on one of the sidewalls which support the cradle shaft.

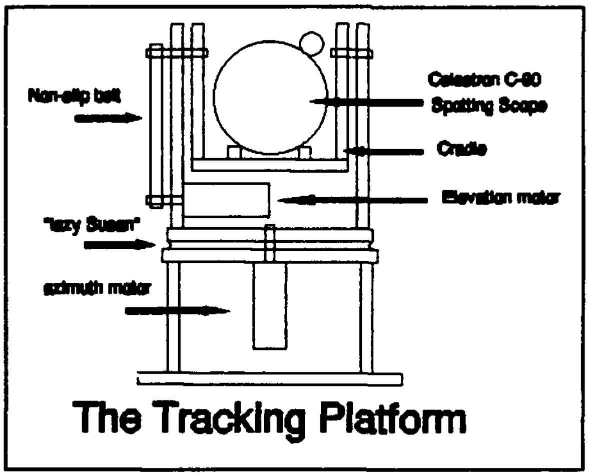


Figure 10

Except for the bearings, the "lazy Susan", the shafts, the motors, and the fastening devices, all of the materials used in the construction of the platform were made of PVC sheet stock. Although PVC is a heavy material, it was selected for use in construction of the tracking platform because it is easy to machine and because it acts as a natural lubricant, which was useful in reducing the friction of the system.

3.2 Signal Collection and Conditioning

The first stage of signal collection was performed by a purchased Celestron C-90 Spotting Scope. The scope, with its

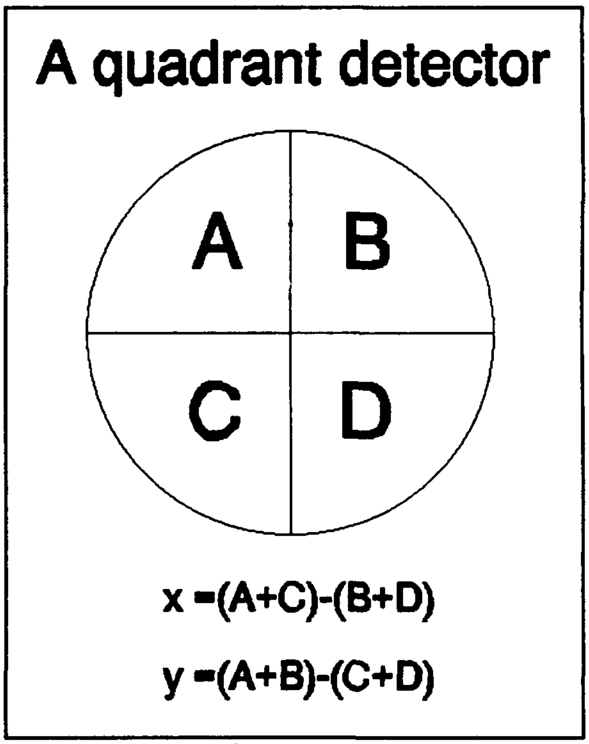


Figure 11

large aperture, was needed to collect red light from the return of a Metrologic laser to produce an image bright enough for the next stage, the photodetector, to detect.

The photodetector used was a United Detector Technologies Spot-9/D quadrant detector. A quadrant detector works by producing a signal in each quadrant (A, B, C, or D, as depicted in Figure 11) that is proportional to the intensity of the light that impinges on each quadrant.[16] As indicated in Figure 11, information can be obtained about the horizontal (x or azimuth) and vertical (y or elevation) positions of the image on a quadrant detector by adding and subtracting the signals obtained from each of the quadrants.[17] However, before addition and subtraction of the electrical signals of the quadrants was performed, several signal conditioning steps had to be taken to produce a useable signal.

The first signal conditioning step for each of the four quadrant detector channels was a conversion of photodetector current to voltage through a transimpedance amplifier. A transimpedance amplifier works by effectively short-circuiting the leads of a photosensitive device to ground, and providing a voltage gain for the resulting current signal.[18]

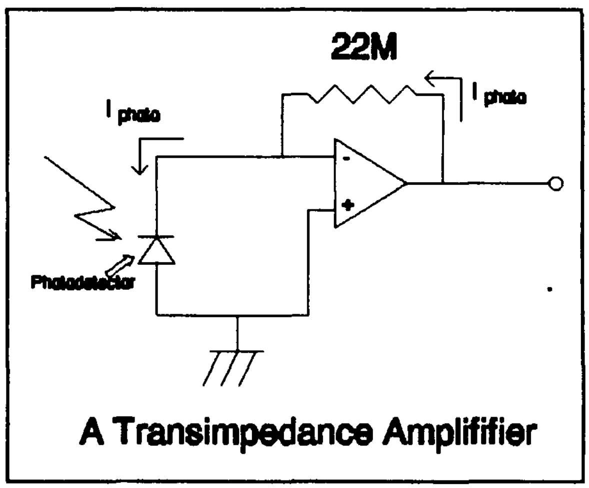


Figure 12

A transimpedance amplifier differs from a normal inverting (gain) amplifier in that an input resistor is not needed for the amplifier, since the photodetector already has a very high internal resistance which serves as the input resistance for the amplifier.

There were two problems in obtaining an adequate signal from the transimpedance amplifier. First, the gain of the transimpedance amplifier required an extremely large feedback resistance, on the order of about 10 megaohms, to provide a recognizable signal. The transimpedance amplifiers used in this project had feedback resistances of 22 megohms, which provided a signal gain of about two.

A second problem with the transimpedance amplifier was a signal bias due to the dark current of the photodetector. This bias was extremely important because it tended to skew the signal significantly. Thus, each quadrant of the photodetector was provided with its own offset in the conditioning circuitry in order to compensate for this bias.

Offset for the dark current of the photodetector was accomplished in a second stage of signal conditioning, which also included a signal gain of five volts/volt. This gain was needed to provide signals large enough to combine in the next steps of the conditioning circuitry, since the signal provided by the transimpedance amplifier was very small (ranging from tenths of millivolts to approximately fifteen millivolts). As mentioned before, each channel had an offset of a few

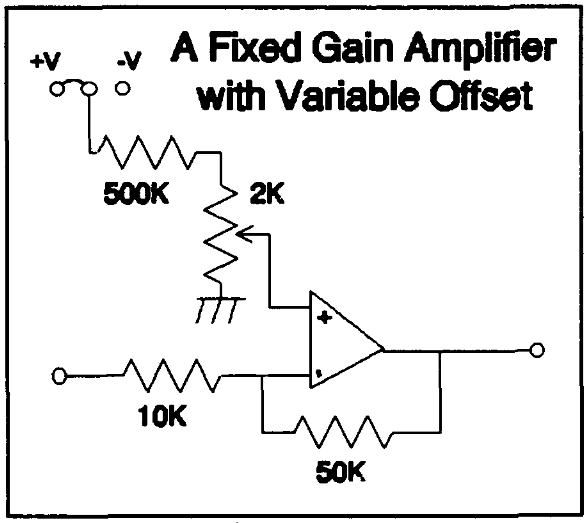


Figure 13

millivolts which tended to bias or, in some cases, completely swamp the desired signal. In order to correct this problem, a variable offset was added to this gain stage.

After these first two signal conditioning steps were performed, the signals, one for each quadrant, were delivered to a series of simple summing and differencing amplifiers to combine the signals into usable azimuth and elevation information. The equations for these summations and subtractions were as follows:

$$azimuth(x) = (A+C)-(B+D)$$

$$elevation(y) = (A+B)-(C+D)$$

These two signal channels were then amplified (through the use of variable-gain amplifiers), and biased to provide signals that ranged from 0 to 5.0 volts. This range was needed to make full use of an analog-to-digital converter, which converted the analog positional information into usable digital form for the fuzzy logic microcontrollers.

The first two signal conditioning stages - transimpedance amplifiers and fixed gain with variable offset, were custom-built on a printed circuit board which was mounted directly on the Celestron spotting scope. This was done to reduce the length of wire leads from the photodetector to these first two critical stages. The remaining stages were built on a generic printed circuit board.

3.3 The Fuzzy Logic Control Algorithm

The fuzzy logic controller was constructed from Reduced InStruction Code Assembly (RISC) Erasable Programmable Read Only Memory (EPROM) microcontrollers. Six microcontrollers were needed for each axis' signal channel, azimuth or elevation. A block diagram describing the flow of information between these microcontrollers is contained in Appendix 9.1.

C language was considered for implementation of the fuzzy logic algorithm, but after experimentation in the earlier stages of the project with C language using an IBM PC, and after conversing with a researcher in the fuzzy logic field[19], it was decided to use RISC-type microcontrollers. C language was deemed much too slow for this high-speed control problem, providing, at best, about 30 iterations (30 Hz) of the control algorithm per second. Although the RISC-type controllers proved to be much harder to program and debug, they were much faster and they allow much greater flexibility because the programmer precisely controlled the flow of information within the algorithm. These RISC-type controllers also provided much greater freedom in the control of timing of program sequences in the algorithm.

As noted before, six microcontrollers were used to provide the control algorithm for each axis' signal channel. One PIC16C71, one PIC16C54, and four PIC16C57 microcontrollers were used for each channel.

The PIC16C71 (encoded with program A2DN02, Appendix 9.2) contained an onboard analog-to-digital (a/d) converter and was used to interface the signal conditioning circuitry with the fuzzy control algorithm. It did this by converting the analog positional signal information into usable digital form for the microcontrollers. The PIC16C71 also computed the derivative of the signal by simple subtraction of two analog-to-digital conversions, one after the other, separated by a controllable time delay. This time delay had the added feature of setting the timing of a complete iteration of the fuzzy logic control algorithm.

One PIC16C57 chip (encoded with program CTRL57, Appendix 9.3) provided control of information from point to point in the algorithm. Its function was to perform all "handshaking" between microcontrollers and to coordinate the flow of information from chip to chip. This was crucial since the number of input/output pins per chip was limited. Thus, this chip ensured that information transactions between microcontrollers occurred only when those microcontrollers were fully ready to send or receive information.

Two more PIC16C57 chips were encoded with information about input membership function domains for each channel. One of these chips (encoded with program POS4, Appendix 9.4) categorized and determined degrees-of-membership for the positional input domain, and the other (encoded with a program cousin to POS4 called VEL4) derived these items for the derivative input domain.

The final PIC16C57 chip (encoded with program LOGIC6, Appendix 9.5) provided rule inferencing functions, including the Fuzzy Associative Matrix and output and weighting functions. The end result from this chip was a single digital motor control signal.

The PIC16C54 chip (encoded with program PWM, Appendix 9.6) changed this digital control signal into a Pulse Width Modulated motor control signal to move an axis' motor. This chip was capable of determining both motor direction and speed from the digital control signal.

In addition to the microcontrollers, a separate digital-to-analog converter chip (an AD558) was used to provide analog information about the output control signal for monitoring purposes.

The fuzzy control algorithm developed in this project could be applied to any one- or two-input control problem with little alteration except for the FAM rule base and the input membership function domains. Currently, the algorithm works with eight-bits of precision. Although greater precision was not needed for this project, the algorithm could easily be expanded to sixteen bits of precision. Expanding the controller to handle another input variable (such as acceleration) could prove to be a difficult task. The concepts used in programming this controller would remain the same, but doing so would expand the rule base from two dimensions to three. The PIC controllers are limited in program and data memory and might not physically be able to handle such a drastic expansion of its responsibilities; the solution of such a problem would at the very least be formidable since the size of rule look-up tables and the size of the program would increase significantly.

One more item should be mentioned about this control algorithm. This algorithm provides the same control as a 64Kbyte look-up table with only 9Kbytes (3000 words) of memory required per channel. Although this comes at a very slight sacrifice for speed, this control algorithm is much more flexible because only 49 rules need to be altered to change system response.

3.31 Analog-to-Digital Conversion - Program A2DNO2

Of the five programs written for the fuzzy logic algorithm, A2DNO2 was one of the simplest. The purpose of this program was to convert analog position information from either axis' signal channel, azimuth or elevation, into usable digital form for the microcontrollers. A few brief notes of information: from this point on, letters in *italics* refer to the names of the routines which perform the functions mentioned in the discussion. Also, a block diagram detailing the flow of information between microcontrollers is included as Appendix 9.1.

After a chip initialization routine (*Initialize*), the program proceeds to a short warm-up delay (*Delay*) to allow the rest of the microcontrollers to catch up and get in synch with each other. The program then performs its first a/d (analog to digital) conversion (*Start1*) and stores the result in a memory register (*Main_loop1*). A counter loop (*Wait1* and *Stop1*) is then entered, and a second a/d conversion is performed (*Start2*), with its result being stored in a second memory register (*Main_loop2*). At this point, the program calculates the derivative of the position by a simple subtraction (*Derivative*). Also, the derivative is multiplied, usually by a factor of four or eight, in order to increase the damping of the system.

The program next moves the result of the second a/d conversion (the most recent position information) onto the output port of the device (*Output_pos*). After "handshaking" with the control chip (*Wait2*), the device then moves the derivative information onto the output port (*Output_vel*). Once more "handshaking" with the control chip is performed (*Wait3*), and the chip enters a second counter loop, identical to the first (*Wait4* and *Stop4*). Together, these two loops determine the frequency (the number of iterations per second) of the control algorithm.

The final step of this program is to jump back to the point where another a/d conversion takes place. The program then starts anew, and fresh data is taken about the position and rate of change of the position with respect to the tracking platform.

3.32 Control of Information - Program CTRL57

A major drawback to the PIC16Cxx family of chips is the limited number of input/output pins which can be used for both control of the chips and transfer of information to and from the chips. This was not a problem for the PIC16C54 and the PIC16C71 microcontrollers used in this system, but it was a problem for all of the PIC16C57 chips, which had a great deal more information to deal with, and which consequently needed more control of the information flow. To alleviate this problem, a special control process (the CTRL57 program) was encoded which allowed a special chip to perform the majority of the data control. This freed up pins on the other chips for the more important tasks of data transfer.

The CTRL57 program can be divided into five major parts. The first part is a brief chip initialization period (*Set_Up*). The next two parts (*Wait1* to *Wait2*, and *Wait3* to *Wait4*) control the flow of digital position and derivative information from the PIC16C71 to the respective input membership function chips. The last two parts (*PosZero* to *HoldASec1*, and *VelZero* to *HoldASec2)* are more complex and deal with the transfer of category and degree-of-membership information from the chips (programmed with POS4 and VEL4) which handle input membership function procedures to the chip (programmed with LOGIC6) which handles rule evaluation procedures. After the CTRL57 program is complete, it loops back to the beginning to initialize another round of data transfer in the control algorithm.

3.33 Input Membership Functions - Programs POS4 and VEL4

POS4 and VEL4 are nearly identical programs used to categorize and determine membership values for (position and derivative, respectively) input information. For the purposes of this paper only POS4 will be discussed.

POS4, like the two programs discussed above, has an initial chip setup routine (*Start*). Once this is complete, the program waits for a "handshake" (*Wait1*) from the control chip, then moves the position information into an input memory register. After another "handshake" with the control chip confirming reception of the information (*Wait2*), the program then moves to routine *Find_Case*, where the program determines where the input falls in the input membership function domain. Thirteen cases are possible. Seven cases (NL,NM,NS,ZE,PS,PM, and PL) place the input completely within (at the peak of) of a membership function. Six more cases (NL&NM, NM&NS, NS&ZE, ZE&PS, PS&PM, and PM&PL) place the input within the domain of two adjoining membership functions. After the case is decided, the program jumps to the case-specific routine which decides how to handle the input information (such as *case_NS:NM*). These routines first assign values to registers that describe which categories the information falls into, and then determine membership values for these categories (through the use of calls to three subroutines, *find_mem1*, *find_mem2*, and *divide*).

At this point in the program, four pieces of information have been derived. This information must now be sent off to the microcontroller which handles the rest of the fuzzy algorithm, and this is accomplished by a series of "handshaking" and data downloading commands (*DataHold* to *WaitThree*). Finally, the program reinitializes and waits for a new set of information in order to start the cycle anew (*Wait*).

3.34 The Fuzzy Logic Control Chip - Program LOGIC6

The LOGIC6 program is the most ambitious of the five created for the fuzzy logic algorithm. This is because it accomplishes the most. The purpose of the LOGIC6 program is to take the eight pieces of information derived from the two input membership function chips and combine them into a single, crisp control output.

As with the other programs, this program begins with an initial chip setup routine (*Start*). After this is complete, the program waits for "handshaking" to occur so it can begin to receive the fuzzified input information from the POS4 and VEL4 input membership function chips (*Move_One* to *Move_Eight*).

Eight bytes of information are taken in by the program. Four bytes describe the categories into which the input information falls (two for positional information and two for derivative information). Four more bytes describe the degree of each category's membership.

After all inputs have been received by the program, the program jumps to a routine which sorts the information and combines it to determine rule values for each unique set of conditions that are fired (*Manipulate*). Since there are two position conditions (categories), and two derivative conditions, the routine must determine four rules. This routine also performs the minimum (***AND***) operation for each of the four rules. The rules themselves are determined by use of a lookup table (*Rules*).

After the rules have been determined, the program jumps to a routine that determines how many unique rules are fired (*Max*). This step is necessary to determine if any fuzzy ***OR*** operations need to be performed for rules that are fired more than once. The *Max* routine accomplishes its task by determining whether or not pairs of rules are equal. After comparing the six possible pairs of rules, the routine is able to distinguish how many different rules have been fired and how the routine needs to proceed to properly combine the rules (*goto_case*). The routine does this by using cases. In all, there are fifteen possible cases under which a combination of rules can fall. Once a case has been invoked, a jump is made to a special routine (such as *case7*) which handles the fuzzy ***OR*** function for that case.

The final step of the program, once all of the rules have been determined and the fuzzy ***ANDs*** and ***ORs*** resolved, is to weight and combine the rules. First, the weights are summed (*Control*). Next, each rule is multiplied (by way of a special multiply function, *Multiply*) by its membership value, and the results are summed together (*_1st_reg* to *_4th_reg*). Finally, the sum of the weighted rules is divided by the sum of the weights, and the final output control value is obtained (*Divide*). The last step of the fuzzy control chip, before it jumps to the beginning to start the process over again, is to send the final control value to the pulse width modulatior chip.

3.35 Pulse Width Modulation - Program PWM

It was decided to use Pulse Width Modulation (PWM) in this Trident application because of its ease of use and because of its precision. Originally, analog motor control was explored, but nonviscous friction in the tracking platform forced the use of Pulse Width Modulated motor control instead.

PWM is the shortest and the simplest of the five microcontroller programs used in this project. The PWM chip achieves its task by taking in the single control output from the fuzzy control chip (*Start*, which occurred after the chip initialization routine, *Set_Up*). The program then determines motor direction from the eighth bit (the sign bit) of this value, and it determines the length of the duty cycle (the time a control voltage is applied to the motor to make the motor move) from the remaining seven bits (*Move* and the *Pulse* and *Rest* routines). The program then jumps back to the beginning to take in a new input and start the process over again.

The output of the PWM microprocessor drives a pulse-width-modulated integrated circuit which provides the power switching needed to move the azimuth and elevation control motors.

Section 4 - System Performance

System performance was measured by experiment. A red Metrologic Helium-Neon laser was projected on a laboratory wall and moved in two dimensions (left-to-right and up-down) by a mirror apparatus and a signal generator. The laser was approximately eighteen feet from the wall, and the lights were turned off to prevent the overhead lights from producing unwanted noise in the photodetector. The analog output (error signal) from the fuzzy controller and the mirror driver signals were recorded using a sampling oscilloscope.

Experiment runs were made once it was proven that the platform could track the target laser. When the platform was first observed to track, the fuzzy logic control algorithm operated at 100Hz. This meant that the system took 100 "snapshots" of the laser image per second, and computed 100 matching control outputs to move the platform. While the platform tracked satisfactorily, the system vibrated violently and regularly lost the laser image. System performance steadily improved (although significant system vibration was still present) when the operating frequency of the platform was increased to 200Hz, then 400Hz, and finally to 600Hz. Increases in operating frequency were stopped at 600Hz because further increases would shorten the time used to obtain the derivative data and would make the derivative data unreliable.

Three plots are given in Appendix 9.7 which illustrate system performance for the azimuth axis of the tracking platform with the system operating at 600Hz. All three plots illustrate an analog representation of the motor control signal. The reason why the motor control or error signal was chosen to demonstrate system performance is that this signal depicts the fuzzy controller's efforts to align the tracking platform with the laser return and this signal it is direct reflection of the tracking platform's error.

The first plot (Appendix 9.71) shows the motor control signal for the tracking platform with no target present. This plot can be considered a representation of the noise present in the tracking platform. The transient spikes in this plot could be from a wide range of sources - ambient light impinging upon the photodetector, noise within the signal conditioning circuitry, noise from the fuzzy microcontroller circuitry, and vibration caused by the pulse width modulated motor control signal. An important fact to notice is that if the signal was averaged, the average would be near zero. This is to be expected, since the platform optics does not have a target which it can follow, and the platform remains stationary.

The second plot (Appendix 9.72) shows the error signal for the tracking platform when the platform is centered on the laser image. The large positive and negative spikes in this plot indicate noise alluded to in the previous plot in addition to a great deal of system vibration. The source of this vibration is most likely the pulse width modulated motor control signals coupled with the extremely sensitive system optics. The platform is constantly trying to reduce its error to zero by aligning itself with the laser image, but in overcoming friction the system often overshoots its target. This overshoot is a problem because even very slight movement of the platform causes significant fluctuation of the laser image's position on the sensitive photodetector. These changes cause the controller to continually overshoot as it attempts to align the platform with the target. Still, however, the control signal averages to zero because the average platform pointing position is centered on the laser image.

The third plot (Appendix 9.73) demonstrates the tracking platform motor control signal as the platform tracks a laser image that sweeps back and forth through an angle of approximately 5.5 degrees on the laboratory wall. The triangular waveform in this plot is the signal which positions the laser mirror. The triangular waveform frequency in this plots is 0.2Hz, which corresponds to a target speed of a little more than two degrees per second. This plot shows that the system is able to track a "slowly moving" target, but, again, that it has enormous problems with vibration. Notice that in this plot that the motor control signal does not average to zero. Instead, on the upward sweep of the triangular waveform (the laser moves left to right), the motor control signal is predominately positive, and on the downward sweep of the triangular waveform (the laser moves right to left), the motor control signal is predominately negative. This means that the controller is pushing the platform in a direction to match the sweep of the laser to keep the platform aligned with the laser image; the tracking platform is tracking the laser.

The maximum speed the laser is able to consistently track at is 3.3 degrees per second. After this point, the laser moves quickly enough that the platform, due to a combination of vibration, friction, and limited field of view, is unable to keep up with the laser image. Although the platform is restricted by the speed with which it can track the laser, its range of operation is almost unlimited. The target projection angles correspond to an azimuth angle of 30 degrees and to an elevation angle of 12 degrees. The platform is able to track the laser image throughout this entire area - until wall space literally runs out.

Section 5 - Future Activities

Presently, the tracking system works well, but it needs several major improvements. The first major area of improvement would be with the tracking platform itself. As designed, the tracking platform, with its high sidewalls that mount the spotting scope cradle, acts like a giant tuning fork. This causes enormous problems with vibration, which tends to limit the speed at which the tracking platform can track its target. This vibration could be limited by redesigning the tracking platform or by incorporating vibration-absorbing materials within the tracking platform.

In addition to vibration, the present tracking platform has a great deal of non-viscous friction which tends to limit the motion of the tracking platform. This was a major motivation for using Pulse Width Modulation motor control - to help overcome this resistance. Friction in the azimuth axis could be reduced by replacing the "lazy Susan" with a sheet Teflon. Friction and vibration in both axes could be reduced by finding better gearing assemblies for the axes' positioning motors.

A second major area of improvement for the tracking system would be to widen the useful field of view of the optical detector. Currently the platform has a useful field of view of only about .75 degrees (both azimuth and elevation). This too, limits the speed at which the platform can track. If the laser image moves too quickly, it can jump out of sensor range before the tracking system can properly respond. The field of view could be widened by using a photodetector with a larger active surface area than the SPOT-9/D used in this project.

A third major area of improvement is the tracking system's signal conditioning circuitry. Currently, the tracking problem is restricted to the use of one type of red Helium-Neon laser. This is because the signal conditioning circuitry was built and tuned to receive information from this one specific type of light energy. Including automatic gain control in the conditioning circuitry would allow the tracking system to track more types of light energy, increasing its flexibility and usefulness as a tracking device.

The fuzzy logic controller hardware could also use improvement. The controller hardware could be improved by using a more powerful microcontroller. The PIC devices used in this project worked well, but they have limitations calling subroutines and determining look-up tables. This required breaking the control algorithm into several pieces and encoding these pieces in separate microcontroller chips. This caused much waste in program space, and this caused much waste in program time for data transfer and control functions between chips.

A final improvement to the tracking system, once the system's other problems are addressed, would be to expand the tracking system's tracking capabilities to three dimensions.

Currently, only two dimensions - azimuth and elevation - are considered. For this to be a full-blown tracking system, the system would also need to be able to track in a third dimension - range. Although it would be difficult to implement, this could be achieved by mounting an optical rangefinder next to the scope and expanding the fuzzy logic controller to handle the third dimension of range. Or, instead of using optical components, radar could be used which would be capable of providing desired azimuth, elevation, and range information.

Section 6 - Conclusion

The objective of the project was met. An optical tracking platform, using fuzzy logic as its means of control, was constructed capable of tracking a target laser in two dimensions, azimuth and elevation. System optics were designed and constructed to receive position and rate of change of position (derivative) information from the target laser in both dimensions. The fuzzy logic controller was built using Reduced Instruction Code (RISC) microcontrollers. When system performance was measured by experiment, it was discovered that the tracking platform was able to track a slowly moving target (at a rate less than or equal to 3.3 degrees per second). Although the platform is limited in the rate at which it can track the laser image, the platform is able to track the image over an almost unlimited range. Major improvements could be made to every major aspect of the tracking system to improve its performance, especially platform tracking speed.

Section 7 - References Cited

1. Zadeh, L.A. "Fuzzy Sets", Information and Control, vol.8, 1965, pp.338-353.

2. Brubaker, David I., and Cedric Sheerer. "Fuzzy-logic system solves control problem", EDN, June 18,1992, p.125.

3. Schwartz, David G., and George J. Klir. "Fuzzy logic flowers in Japan", IEEE Spectrum, July 1992, p.34.

4. Brubaker, David I., and Cedric Sheerer. "Fuzzy-logic system solves control problem", EDN, June 18,1992, p.125.

5. Interview with Professor Fuller, Electrical Engineering Department, University of Missouri-Columbia, November 12, 1994.

6. Pacini, Peter J. and Bart Kosko. "Comparison of Fuzzy and Kalman-Filter Target-Tracking Control Systems", Neural Networks and Fuzzy Systems, Prentice-Hall, Inc.: Englewood Cliffs, NJ, 1992, pp.379-406.

7. Gaines, Brian R. "Precise past-fuzzy future", International Journal of Man-Machine Studies, vol.19, 1983, p.120

8. Zadeh, L.A. "Probability Measures of Fuzzy Events", Journal of Mathematical Analysis and Applications, vol.23, 1968, p.421.

9. Kosko, Bart. "Fuzziness Versus Probability", Neural Networks and Fuzzy Systems, Prentice-Hall, Inc., Englewood Cliffs, NJ: 1992, p.265.

10. Li, Y.F. and C.C. Lau. "Development of Fuzzy Algorithms for Servo Systems", IEEE Control Systems Magazine, April 1989, p.65.

11. Li, Y.F., and C.C. Lau. "Development of Fuzzy Algorithms for Servo Systems", IEEE Control Systems Magazine, April 1989, p.65.

12. Whalen, Thomas, and Brian Schott. "Issues in Fuzzy Production Systems", International Journal of Man-Machine Studies, vol.19, 1983, p.57.

13. Zadeh, L.A. "Fuzzy Sets", Information and Control, vol.8, 1965, p.340.

14. Zadeh, L.A. "Fuzzy Sets", Information and Control, vol.8, 1965, p.341.

15. Sibigroth, James M. "Implementing Fuzzy Expert Rules in Hardware", AI Expert, April 1992, p.3. (Motorola Corp. Reprint)

16. Edwards, Ian. "Using Photodetectors for Position Sensing", Sensors, December 1988.

17. Edwards, Ian. "Using Photodetectors for Position Sensing", Sensors, December 1988.

18. "Chapter 22-Detectors", Optics Guide 5, Melles Griot: Irvine, CA, 1992, p.22-19.

19. Interview with Assistant Professor Nair, University of Missouri-Columbia, November 12, 1993.

Section 8 - Bibliographyy

Anderson, Glenn. "Applying Fuzzy Logic In the Real World", Sensors, September 1992, pp.15-25.

Bandler, Wyllis and Ladislav Kohout. "Fuzzy Power Sets and Fuzzy Implication Operators", Fuzzy Sets and Systems, vol.4., 1980, pp.13-30.

Berenji, Hamid R. and Pratap Khedkar. "Learning and Tuning Fuzzy Logic Controllers Through Reinforcements", IEEE Transactions on Neural Networks, vol.3, no.5, September 1992, pp.724-740.

Bernard, John A. "Use of a Rule-Based System for Process Control", IEEE Control Systems Magazine, October 1988, pp.3-13.

Black, Max. "Vagueness-An Exercise in Logical Analysis", Philosophy of Science, vol.4, 1937, pp.427-455.

Brubaker, David I and Cedric Sheerer. "Fuzzy-Logic System Solves Control Problem", EDN, June 18,1992, pp.121-127.

"Chapter 22-Detectors", Optics Guide 5, Melles Griot, Irvine, CA, 1992, pp.22-1 to 22-4.

Edwards, Ian. "Using Photodetectors for Position Sensing", Sensors, December 1988.

Freeling, Anthony N.S. "Fuzzy Sets and Decision Analysis", IEEE Transactions on Systems, Man, and Cybernetics, vol.SMC-10, no.7, July 1980, pp.341-354.

Gaines, Brian R. "Foundations of Fuzzy Reasoning", International Journal of Man-Machine Studies, vol.8, 1976, pp.623-668.

Gaines, Brian R. "Precise Past-Fuzzy Future", International Journal of Man-Machine Studies, vol.19, 1983, pp.117-134.

Interview with Professor Fuller, Electrical Engineering Department, University of Missouri-Columbia, November 12, 1993.

Interview with Assistant Professor Nair, Aerospace and Mechanical Engineering Department, University of Missouri-Columbia, November 12, 1993.

Kong, Seong-Gon and Bart Kosko. "Comparison of Fuzzy and Neural Truck Backer-Upper Control Systems", Neural Networks and Fuzzy Systems, Prentice Hall, Englewood Cliffs, New Jersey, 1992, pp. 339-361.

Kosko, Bart. "Fuzzy Entropy and Conditioning", Information Sciences, vol.40, 1986, pp.165-174.

Kosko, Bart. "Fuzzy Knowledge Combination", International Journal of Intelligent Systems, vol.1, 1986, pp.293-320.

Li, Y.F. and C.C.Lau. "Development of Fuzzy Algorithms for Servo Systems", IEEE Control Systems Magazine, April 1989, pp.65-71.

Pacini, Peter J. and Bart Kosko. "Adaptive Fuzzy Systems for Target Tracking", Intelligent Systems Engineering, vol.1, no.1, Autumn 1992, pp.3-20.

Schwartz, Daniel G. and George J. Klir. "Fuzzy Logic Flowers in Japan", IEEE Spectrum, July 1992, pp.32-35.

Sibigroth, James M. "Implementing Fuzzy Expert Rules in Hardware", AI Expert, April 1992, 6pp. (Motorola Corp. Reprint)

Whalen, Thomas and Brian Schott. "Issues in Fuzzy Production Systems", International Journal of Man-Machine Studies, vol.19, 1983, pp.57-71.

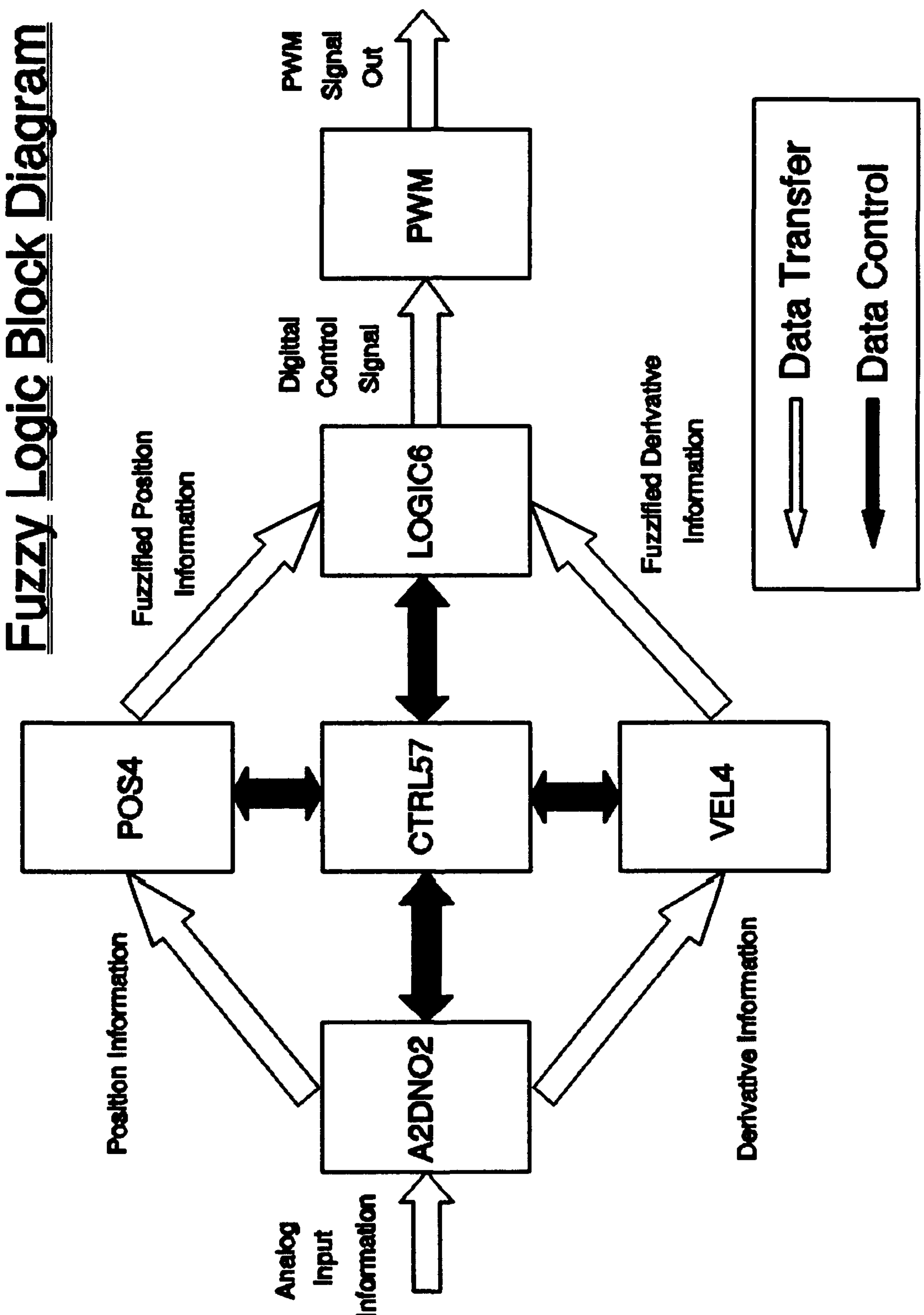
Zadeh, L.A. "Fuzzy Sets", Information and Control, vol.8, 1965, pp.338-353.

Zadeh, L.A. "Fuzzy Sets as a Basis for a Theory of Possibility", Fuzzy Sets and Systems, vol.1, 1978, North-Holland Publishing Company, pp.3-28.

Zadeh, L.A. "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes", IEEE Transactions on Systems, Man, and Cybernetics, vol.SMC-3, no.1, January 1973, pp.28-44.

Zadeh, L.A. "Probability Measures of Fuzzy Events", Journal of Mathematical Analysis and Applications, vol.23, 1968, pp.421-427.

Appendix 9.1 - Program Block Diagram



Appendix 9.2 - Program A2DNO2

```
;Program  a2dno2.src

           DEVICE HS_OSC,WDT_OFF,PWRT_ON,PROTECT_OFF
           ID 'TEST'

value1     equ    0Ch
value2     equ    0Dh
deriv      equ    0Eh
count1     equ    0Fh
count2     equ    10h
count      equ    11h
count3     equ    12h

           org    00h

Initialize setb   RP0                    ;upper register page
           mov    TRISA,#0111b           ;first two bits of port A are
                                         ;analog inputs, other two bits
                                         ;are digital inputs/outputs
           clrb   RA.3
           mov    TRISB,#00000000b       ;port B output
           clr    RB
           clrb   PCFG0                  ;ADCON1.1 - RA.0 & RA.1 are
           setb   PCFG1                  ;ADCON1.0 - analog inputs
                                         ;RA.2 & RA.3 are digital
                                         ;Vdd Ref
           clrb   RP0                    ;lower register page
           clrb   GIE                    ;global interrupt enable cleared -
                                         ;disables all interrupts
           clr    ADCON0                 ;start off with all zeros

           clr    count1
           mov    count2,#10h
Delay      djnz   count1,Delay
           djnz   count2,Delay

;first a/d conversion
;ADCON0 - bit 0 is enable, bit 1 is interrupt, bit 2 is GO/DONE bit, bits
;3&4 decide which channel is analog input, bit 5 is storage, bits 6&7
;select A/D conversion clock source

Start1     setb   ADCON0.0               ;turn a/d on
           clrb   ADCON0.1
           clrb   ADCON0.2               ;AIN0 is input channel
           setb   ADCON0.6
           setb   ADCON0.7               ;use on-chip RC oscillator
           setb   ADCON0.2               ;turn on a/d converter

Main_loop1 jb     ADCON0.2,Main_loop1 ;wait for conversion
           mov    value1,ADRES           ;put result into value
           clr    ADCON0

;delay for second a/d conversion

Wait1      clr    count1
           mov    count2,#02h

Stop1      djnz   count1,Stop1
           djnz   count2,Stop1

;second a/d conversion

Start2     setb   ADCON0.0               ;turn a/d on
           clrb   ADCON0.1
           clrb   ADCON0.2               ;AIN0 is input channel
           setb   ADCON0.6
```

```
            setb    ADCON0.7            ;use on-chip RC oscillator
            setb    ADCON0.2            ;turn on a/d converter

Main_loop2  jb      ADCON0.2,Main_loop2  ;wait for conversion
            mov     value2,ADRES         ;put result into value
            clr     ADCON0

Derivative  mov     deriv,value2
            sub     deriv,value1        ;may need to do some more with deriv
            clc
            rl      deriv
            clc
            rl      deriv               ;multiple deriv by eight!!!
            add     deriv,#10000000b    ;"bias" to zero

Output_pos  mov     RB,value2
            setb    RA.3

            clr     count1

Wait2       jnb     RA.2,Wait2

Output_vel  mov     RB,deriv
            clrb    RA.3

Wait3       jb      RA.2,Wait3

Wait4       clr     count1
            mov     count2,#02h

Stop4       djnz    count1,Stop4
            djnz    count2,Stop4

            jmp     Start1
```

Appendix 9.3 - Program CTRL57

```
;15 Mar 94
;Program for control 57 - Program ctrl57.src

          DEVICE PIC16C57,HS_OSC,WDT_OFF,PROTECT_OFF

          RESET Set_Up

count         equ     08h
count1        equ     09h
count2        equ     0Ah


Set_Up        mov     !RA,#1100b       ;control of pos chip
              mov     !RB,#11101100b  ;control of derv chip, pic71, fuzzy chip
              mov     !RC,#00001100b

;control of position information from PIC71 to input mem fn chip

Wait1         clr     RA
              clr     RB
              clr     RC
              jnb     RB.5,Wait1
              setb    RA.0
Wait2         jnb     RA.2,Wait2

;control of derivative information from PIC71 to input mem fn chip

              setb    RB.4
Wait3         jb      RB.5,Wait3
              setb    RB.0
Wait4         jnb     RB.2,Wait4

;preset for output

              setb    RA.1
              setb    RB.1

;release a/d converter

              clrb    RB.4


;control of position input mem fn data to fuzzy chip

              setb    RA.0
              clrb    RA.1
PosZero       jb      RA.2,PosZero
              jb      RA.3,PosZero
              setb    RC.0
              clrb    RC.1
FuzPosZero    jb      RC.2,FuzPosZero
              jb      RC.3,FuzPosZero
              clrb    RA.0
              clrb    RA.1

PosOne        jnb     RA.2,PosOne
              jb      RA.3,PosOne
              clrb    RC.0
              setb    RC.1
FuzPosOne     jnb     RC.2,FuzPosOne
              jb      RC.3,FuzPosOne
              setb    RA.0
              clrb    RA.1

PosTwo        jb      RA.2,PosTwo
              jnb     RA.3,PosTwo
              setb    RC.0
```

```
            setb    RC.1
FuzPosTwo   jb      RC.2,FuzPosTwo
            jnb     RC.3,FuzPosTwo
            clrb    RA.0
            setb    RA.1

PosThree    jnb     RA.2,PosThree
            jnb     RA.3,PosThree
            clrb    RC.0
            clrb    RC.1
FuzPosThree jnb     RC.2,FuzPosThree
            jnb     RC.3,FuzPosThree
            setb    RA.0
            setb    RA.1

            mov     count,#10h
HoldASec1   djnz    count,HoldASec1
            clrb    RA.0
            setb    RA.1

;control of velocity input mem fn data to fuzzy chip

            setb    RB.0
            clrb    RB.1
VelZero     jb      RB.2,VelZero
            jb      RB.3,VelZero
            setb    RC.0
            clrb    RC.1
FuzVelZero  jb      RC.2,FuzVelZero
            jb      RC.3,FuzVelZero
            clrb    RB.0
            clrb    RB.1

VelOne      jnb     RB.2,VelOne
            jb      RB.3,VelOne
            clrb    RC.0
            setb    RC.1
FuzVelOne   jnb     RC.2,FuzVelOne
            jb      RC.3,FuzVelOne
            setb    RB.0
            clrb    RB.1

VelTwo      jb      RB.2,VelTwo
            jnb     RB.3,VelTwo
            setb    RC.0
            setb    RC.1
FuzVelTwo   jb      RC.2,FuzVelTwo
            jnb     RC.3,FuzVelTwo
            clrb    RB.0
            setb    RB.1

VelThree    jnb     RB.2,VelThree
            jnb     RB.3,VelThree
            clrb    RC.0
            clrb    RC.1
FuzVelThree jnb     RC.2,FuzVelThree
            jnb     RC.3,FuzVelThree
            setb    RB.0
            setb    RB.1

            mov     count,#10h
HoldASec2   djnz    count,HoldASec2
            clrb    RB.0
            setb    RB.1

            ljmp    Wait1
```

Appendix 9.4 - Program POS4

```
;Program pos4.src
;input membership function domian for position

                DEVICE PIC16C57,HS_OSC,WDT_OFF,PROTECT_OFF

                RESET Start

register_1      equ     08h
mem_reg_1       equ     09h
register_2      equ     0Ah
mem_reg_2       equ     0Bh

NL              equ     0Ch
NM              equ     0Dh
NS              equ     0Eh
ZE              equ     0Fh
PS              equ     10h
PM              equ     11h
PL              equ     12h

Ldist           equ     13h
Mdist           equ     14h
Sdist           equ     15h
ZEdist          equ     16h

position        equ     17h

l_value         equ     18h
r_value         equ     19h

number1         equ     1Ah
number2         equ     1Bh

upper_num       equ     10h       ; (page 1)?!
lower_num       equ     11h
upper_div       equ     12h
lower_div       equ     13h
divdiv2         equ     15h
counter1        equ     16h
counter2        equ     17h
answer          equ     18h
number          equ     19h
divisor         equ     14h       ; (same as divisor)

Start           mov     !RA,#0011b
                mov     !RB,#11111111b
                mov     !RC,#11111111b
                clr     RA
                clr     RB
                clr     RC

                mov     NL,#32
                mov     NM,#64
                mov     NS,#96
                mov     ZE,#128
                mov     PS,#160
                mov     PM,#192
                mov     PL,#224

                mov     Ldist,#32
                mov     Mdist,#32
                mov     Sdist,#32
                mov     ZEdist,#32

Wait1           clr     RA  ;input handshake
                clr     RB
```

```
                clr     RC
                jnb     RA.0,Wait1
                mov     position,RB
                setb    RA.2
Wait2           jnb     RA.1,Wait2
                setb    RA.3        ;preset for output

;determine where the data falls in the input membership domain

Find_Case       cjae    position,PL,case_PL
                cjbe    position,NL,case_NL
                cja     position,PM,case_PM:PL
                cjb     position,NM,case_NM:NL
                cje     position,PM,case_PM
                cje     position,NM,case_NM
                cja     position,PS,case_PS:PM
                cjb     position,NS,case_NS:NM
                cje     position,PS,case_PS
                cje     position,NS,case_NS
                cja     position,ZE,case_ZE:PS
                cjb     position,ZE,case_ZE:NS
                cje     position,ZE,case_ZE

case_NL         mov     register_1,#0001b
                ljmp    B_4_Next_Step

case_NM         mov     register_1,#0010b
                ljmp    B_4_Next_Step

case_NS         mov     register_1,#0011b
                ljmp    B_4_Next_Step

case_ZE         mov     register_1,#0100b
                ljmp    B_4_Next_Step

case_PS         mov     register_1,#0101b
                ljmp    B_4_Next_Step

case_PM         mov     register_1,#0110b
                ljmp    B_4_Next_Step

case_PL         mov     register_1,#0111b
                ljmp    B_4_Next_Step

case_NM:NL      mov     register_1,#0010b
                mov     register_2,#0001b
                mov     number1,position
                mov     l_value,NM
                sub     l_value,Mdist
                movf    Mdist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                lcall   find_mem1
                mov     number2,NL
                add     number2,Ldist
                movf    Ldist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                ljmp    find_mem2

case_NS:NM      mov     register_1,#0011b
                mov     register_2,#0010b
                mov     number1,position
                mov     l_value,NS
                sub     l_value,Sdist
```

```
                movf    Sdist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                lcall   find_mem1
                mov     number2,NM
                add     number2,Mdist
                movf    Mdist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                ljmp    find_mem2

case_ZE:NS      mov     register_1,#0100b
                mov     register_2,#0011b
                mov     number1,position
                mov     l_value,ZE
                sub     l_value,ZEdist
                movf    ZEdist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                lcall   find_mem1
                mov     number2,NS
                add     number2,Sdist
                movf    Sdist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                ljmp    find_mem2

case_ZE:PS      mov     register_1,#0101b
                mov     register_2,#0100b
                mov     number1,position
                mov     l_value,PS
                sub     l_value,Sdist
                movf    Sdist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                lcall   find_mem1
                mov     number2,ZE
                add     number2,ZEdist
                movf    ZEdist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                ljmp    find_mem2

case_PS:PM      mov     register_1,#0110b
                mov     register_2,#0101b
                mov     number1,position
                mov     l_value,PM
                sub     l_value,Mdist
                movf    Mdist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                lcall   find_mem1
                mov     number2,PS
                add     number2,Sdist
                movf    Sdist,0
```

```
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                ljmp    find_mem2

case_PM:PL      mov     register_1,#0111b
                mov     register_2,#0110b
                mov     number1,position
                mov     l_value,PL
                sub     l_value,Ldist
                movf    Ldist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                lcall   find_mem1
                mov     number2,PM
                add     number2,Mdist
                movf    Mdist,0
                clrb    04h.6
                setb    04h.5
                movwf   divisor
                mov     04h,#00000000b
                ljmp    find_mem2

                org     200h

find_mem1       sub     number1,l_value
                movf    number1,0
                clrb    04h.6
                setb    04h.5
                movwf   number
                lcall   divide
                movf    answer,0
                clrb    04h.6
                clrb    04h.5
                movwf   mem_reg_1
                mov     04h,#00000000b
                bcf     3,5
                bcf     3,6
                ret

find_mem2       sub     number2,position
                movf    number2,0
                clrb    04h.6
                setb    04h.5
                movwf   number
                lcall   divide
                movf    answer,0
                clrb    04h.6
                clrb    04h.5
                movwf   mem_reg_2
                mov     04h,#00000000b
                jmp     DataHold

B_4_Next_Step   mov     mem_reg_1,#11111111b
                mov     register_2,#0000b
                clr     mem_reg_2

;output handshake and data download

DataHold        jnb     RA.0,DataHold
                jb      RA.1,DataHold
                mov     !RC,#00000000b
Zero            mov     RC,register_1
                clrb    RA.2
                clrb    RA.3
```

```
WaitZero        jb      RA.0,WaitZero
                jb      RA.1,WaitZero
One             mov     RC,mem_reg_1
                setb    RA.2
                clrb    RA.3

WaitOne         jnb     RA.0,WaitOne
                jb      RA.1,WaitOne
Two             mov     RC,register_2
                clrb    RA.2
                setb    RA.3

WaitTwo         jb      RA.0,WaitTwo
                jnb     RA.1,WaitTwo
Three           mov     RC,mem_reg_2
                setb    RA.2
                setb    RA.3

WaitThree       jnb     RA.0,WaitThree
                jnb     RA.1,WaitThree

                clr     RC              ;still a problem here!
                mov     !RC,#11111111b

                setb    RA.2
                clrb    RA.3
Wait            jb      RA.0,Wait
                jnb     RA.1,Wait

                ljmp    Wait1

                org     400h

divide          mov     04h,#00100000b
                clr     counter1                ;clear counter1
                clr     counter2                ;clear counter2
                clr     answer                  ;clear answer
                mov     divdiv2,divisor         ;move number __ into divdiv2
                clc                             ;clear carry
                rr      divdiv2                 ;rotate divdiv2 one bit to right (/2)
                addb    divdiv2,c               ;add c nit to dividiv2
                mov     upper_num,number        ;multiply number by 256
                mov     upper_div,divisor       ;mov divisor into upper byte
                clr     lower_num               ;clear lower byte of number
                clr     lower_div               ;clear lower byte of divisor
                clc                             ;clear carry
                jmp     count_zeros ;lcall?ljump?jump???  ;call subroutine to count zeros
back            add     counter1,#00001000b     ;add 8 to counter1
                jmp     long_div                ;jump to long_div

count_zeros     snb     upper_div.7             ;if 7th bit is "1", then return
                jmp     back                    ;(return)
                rl      upper_div               ;shift divisor one bit to left
                inc     counter1                ;and one to zeros counter
                jmp     count_zeros             ;check next bit

long_div        cja     upper_div,upper_num,comp_counters
                cje     upper_div,upper_num,upper_equal
subtract        sub     lower_num,lower_div             ;subtract divisor from number
                sc
                dec     upper_num
                sub     upper_num,upper_div
                inc     answer                          ;add one to answer
comp_counters   cje     counter2,counter1,Remainder     ;if shifted to right as many zeros as
shifted to left, go to Output
                inc     counter2                        ;add one to counter2
                clc                             ;clear carry register
                rl      answer                  ;shift answer one bit to left
                clc
```

```
                rr      upper_div               ;shift divisor  one bit to right
                rr      lower_div
                jmp     long_div                ;jump to long_div

upper_equal     cja     lower_div,lower_num,comp_counters
                jmp     subtract

Remainder       cjb     lower_num,divdiv2,Done_Division ;see if remainder can be rounded
                add     answer,#00000001b               ;if so, then add one to answer

Done_Division   bsf     3,5
                bcf     3,6
                ret
```

Appendix 9.5 - LOGIC6

```
;Program - logic6.src
;fuzzy logic control chip

                DEVICE PIC16C57,HS_OSC,WDT_OFF,PROTECT_OFF

                RESET Start

position1       equ     08h     ;input registers
position2       equ     09h     ; "
velocity3       equ     0Ah     ; "
velocity4       equ     0Bh     ; "

mempos1         equ     0Ch     ; "
mempos2         equ     0Dh     ; "
memvel3         equ     0Eh     ; "
memvel4         equ     0Fh     ; "

register1       equ     10h
register2       equ     11h
register3       equ     12h
register4       equ     13h

member1         equ     14h     ;needed for final step
member2         equ     15h     ; "
member3         equ     16h     ; "
member4         equ     17h     ; "

rule1           equ     18h     ; "
rule2           equ     19h     ; "
rule3           equ     1Ah     ; "
rule4           equ     1Bh     ; "

member          equ     08h
register        equ     09h
rule            equ     0Ah
case            equ     0Bh

answer_upper    equ     0Bh     ;these will be used for the intermedite
answer_lower    equ     14h     ;step of weighting before the final control
                                ;output is obtained
reg             equ     17h     ;holds intermediate value of reg(1,2,3,4)
val             equ     18h     ;ditto, except for val(1,2,3,4)
counter         equ     19h     ;counter for 8-bits

reg1            equ     0Ch     ;these are the input function max values
reg2            equ     0Dh
reg3            equ     0Eh
reg4            equ     0Fh

val1            equ     10h     ;these are the control rule values
val2            equ     11h     ;taken from the FAM rule base
val3            equ     12h
val4            equ     13h

weight_low      equ     1Ch     ;low byte of the weight (divisor)
weight_high     equ     1Dh     ;high byte of the weight (divisor)

control_high    equ     1Eh     ;high byte of the control
control_low     equ     1Fh     ;low byte of control

counter1        equ     08h     ;counts # of shifts left
counter2        equ     09h     ;counts # of shifts right
upper_divdiv2   equ     10h
lower_divdiv2   equ     11h
answer          equ     12h
```

```
;*******************************************************************************

                org     000h

Start           mov     !RA,#0011b
                mov     !RB,#11111111b
                mov     !RC,#00000000b

;handshaking and input function

Move_One        clr     RA
                clr     RB
                setb    RA.2
                setb    RA.3
                jnb     RA.0,Move_One
                jb      RA.1,Move_One
                nop
                mov     position1,RB
                clrb    RA.2
                clrb    RA.3

Move_Two        jb      RA.0,Move_Two
                jnb     RA.1,Move_Two
                nop
                mov     mempos1,RB
                setb    RA.2
                clrb    RA.3

Move_Three      jnb     RA.0,Move_Three
                jnb     RA.1,Move_Three
                nop
                mov     position2,RB
                clrb    RA.2
                setb    RA.3

Move_Four       jb      RA.0,Move_Four
                jb      RA.1,Move_Four
                nop
                mov     mempos2,RB
                setb    RA.2
                setb    RA.3

Move_Five       jnb     RA.0,Move_Five
                jb      RA.1,Move_Five
                nop
                mov     velocity3,RB
                clrb    RA.2
                clrb    RA.3

Move_Six        jb      RA.0,Move_Six
                jnb     RA.1,Move_Six
                nop
                mov     memvel3,RB
                setb    RA.2
                clrb    RA.3

Move_Seven      jnb     RA.0,Move_Seven
                jnb     RA.1,Move_Seven
                nop
                mov     velocity4,RB
                clrb    RA.2
                setb    RA.3

Move_Eight      jb      RA.0,Move_Eight
                jb      RA.1,Move_Eight
                nop
                mov     memvel4,RB
                setb    RA.2
                setb    RA.3
```

```
                ljmp    Manipulate

;******************************************************************************

                org     200h

Rules           jmp     PC+W

                retw    128,128,128,128,128,128,128,128
                retw    128,  0, 21, 43, 64, 85,106,128
                retw    128, 21, 43, 64, 85,106,128,149
                retw    128, 43, 64, 85,106,128,149,171
                retw    128, 64, 85,106,128,149,171,192
                retw    128, 85,106,128,149,171,192,213
                retw    128,106,128,149,171,192,213,234
                retw    128,128,149,171,192,213,234,255

;******************************************************************************

;determine rule call numbers for membership pairs

Manipulate      clc

                rl      velocity3
                rl      velocity3
                rl      velocity3
                rl      velocity4
                rl      velocity4
                rl      velocity4

                mov     register1,position1     ;most positive of positions
                add     register1,velocity3     ;most positive of velocities

                mov     register2,position1     ;most positive of positions
                add     register2,velocity4     ;most negative of velocities

                mov     register3,position2     ;most negative of positions
                add     register3,velocity3     ;most positive of velocities

                mov     register4,position2     ;most negative of positions
                add     register4,velocity4     ;most negative of velocities

                mov     member1,mempos1
                cjbe    member1,memvel3,One
                mov     member1,memvel3
One             mov     W,register1
                call    Rules                   ;goto rules
                mov     rule1,W                 ;control value for the rule

                mov     member2,mempos1
                cjbe    member2,memvel4,Two
                mov     member2,memvel4
Two             mov     W,register2
                call    Rules                   ;goto rules
                mov     rule2,W                 ;control value for the rule

                mov     member3,mempos2
                cjbe    member3,memvel3,Three
                mov     member3,memvel3
Three           mov     W,register3
                call    Rules
                mov     rule3,W

                mov     member4,mempos2
                cjbe    member4,memvel4,Four
                mov     member4,memvel4
Four            mov     W,register4
                call    Rules
                mov     rule4,W
```

```
                ljmp    Max

;*****************************************************************************

                org     400h

Max             clrb    3.5
                setb    3.6
                clr     case
                clr     reg2    ;don't need to clear reg1 or val1 since
                clr     reg3    ;at very least these two will be used
                clr     reg4
                clr     val2
                clr     val3
                clr     val4

                csne    rule1,rule2
                setb    case.0
                csne    rule1,rule3
                setb    case.1
                csne    rule2,rule4
                setb    case.2
                csne    rule3,rule4
                setb    case.3
                csne    rule1,rule4     ;adding these will give
                setb    case.4          ;me more flexibility
                csne    rule2,rule3
                setb    case.5

goto_case       cje     case,#48,case13
                cje     case,#16,case14
                cje     case,#32,case15

                clrb    case.4
                clrb    case.5

                mov     W,case

                jmp     PC+W

                jmp     case12      ;#00000000b  0    ;4 rules - 1,2,3,4
                jmp     case11      ;#00000001b  1    ;3 rules - 1&2,3,4
                jmp     case10      ;#00000010b  2    ;3 rules - 1&3,2,4
                jmp     case6       ;#00000011b  3    ;2 rules - 1&2&3,4
                jmp     case8       ;#00000100b  4    ;3 rules - 2&4,1,3
                jmp     case7       ;#00000101b  5    ;2 rules - 1&2&4,3
                jmp     case3       ;#00000110b  6    ;2 rules - 1&3,2&4
                nop
                jmp     case9       ;#00001000b  8    ;3 rules - 1,2,3&4
                jmp     case2       ;#00001001b  9    ;2 rules - 1&2,3&4
                jmp     case5       ;#00001010b  10   ;2 rules - 1&3&4,2
                nop
                jmp     case4       ;#00001100b  12   ;2 rules - 2&3&4,1
                nop
                nop
                jmp     case1       ;#00001111b  15   ;1 rule - 1&2&3&4


case1           mov     reg1,member1    ;1 rule
                cjae    reg1,member2,The
                mov     reg1,member2
The             cjae    reg1,member3,Quick
                mov     reg1,member3
Quick           cjae    reg1,member4,Brown
                mov     reg1,member4
Brown           mov     val1,rule1
                ljmp    Control

case2           mov     reg1,member1    ;2 rules - 1&2,3&4
```

```
        cjae    reg1,member2,Fox
        mov     reg1,member2
Fox     mov     reg2,member3
        cjae    reg2,member4,Jumped
        mov     reg2,member4
Jumped  mov     val1,rule1
        mov     val2,rule3
        ljmp    Control

case3   mov     reg1,member1    ;2 rules - 1&3,2&4
        cjae    reg1,member3,Over
        mov     reg1,member3
Over    mov     reg2,member2
        cjae    reg2,member4,The2
        mov     reg2,member4
The2    mov     val1,rule1
        mov     val2,rule2
        ljmp    Control

case4   mov     reg1,member1    ;2 rules - 1,2&3&4
        mov     reg2,member2
        cjae    reg2,member3,Lazy
        mov     reg2,member3
Lazy    cjae    reg2,member4,Little
        mov     reg2,member4
Little  mov     val1,rule1
        mov     val2,rule2
        ljmp    Control

case5   mov     reg1,member2    ;2 rules - 2,1&3&4
        mov     reg2,member1
        cjae    reg2,member3,Dog
        mov     reg2,member3
Dog     cjae    reg2,member4,Which
        mov     reg2,member4
Which   mov     val1,rule2
        mov     val2,rule1
        ljmp    Control

case6   mov     reg1,member4    ;2 rules - 4,1&2&3
        mov     reg2,member1
        cjae    reg2,member2,Had
        mov     reg2,member2
Had     cjae    reg2,member3,The3
        mov     reg2,member3
The3    mov     val1,rule4
        mov     val2,rule1
        ljmp    Control

case7   mov     reg1,member3    ;2 rules - 3,1&2&4
        mov     reg2,member1
        cjae    reg2,member2,Cutest
        mov     reg2,member2
Cutest  cjae    reg2,member4,Pointy
        mov     reg2,member4
Pointy  mov     val1,rule3
        mov     val2,rule1
        ljmp    Control

case8   mov     reg1,member1    ;3 rules - 1,3,2&4
        mov     reg2,member3
        mov     reg3,member2
        cjae    reg3,member4,Ears
        mov     reg3,member4
Ears    mov     val1,rule1
        mov     val2,rule3
        mov     val3,rule2
        ljmp    Control
```

```
case9       mov     reg1,member1    ;3 rules - 1,2,3&4
            mov     reg2,member2
            mov     reg3,member3
            cjae    reg3,member4,That
            mov     reg3,member4
That        mov     val1,rule1
            mov     val2,rule2
            mov     val3,rule3
            ljmp    Control

case10      mov     reg1,member2    ;3 rules - 2,4,1&3
            mov     reg2,member4
            mov     reg3,member1
            cjae    reg3,member3,You
            mov     reg3,member3
You         mov     val1,rule2
            mov     val2,rule4
            mov     val3,rule1
            ljmp    Control

case11      mov     reg1,member3    ;3 rules - 3,4,1&2
            mov     reg2,member4
            mov     reg3,member1
            cjae    reg3,member2,Could
            mov     reg3,member2
Could       mov     val1,rule3
            mov     val2,rule4
            mov     val3,rule1
            ljmp    Control

case12      mov     reg1,member1    ;4 rules -1,2,3,4
            mov     reg2,member2
            mov     reg3,member3
            mov     reg4,member4
            mov     val1,rule1
            mov     val2,rule2
            mov     val3,rule3
            mov     val4,rule4
            ljmp    Control

case13      mov     reg1,member1   ;2 rules - 1&4,2&3
            cjae    reg1,member4,Ever
            mov     reg1,member4
Ever        mov     reg2,member2
            cjae    reg2,member3,Possibly
            mov     reg2,member3
Possibly    mov     val1,rule1
            mov     val2,rule2
            ljmp    Control

case14      mov     reg1,member1   ;3 rules - 1&4,2,3
            cjae    reg1,member4,Imagine
            mov     reg1,member4
Imagine     mov     reg2,member2
            mov     reg3,member3
            mov     val1,rule1
            mov     val2,rule2
            mov     val3,rule3
            ljmp    Control

case15      mov     reg1,member2    ;3 rules - 2&3,1,4
            cjae    reg1,member3,Dude
            mov     reg1,member3
Dude        mov     reg2,member1
            mov     reg3,member4
            mov     val1,rule2
            mov     val2,rule1
            mov     val3,rule4
            ljmp    Control
```

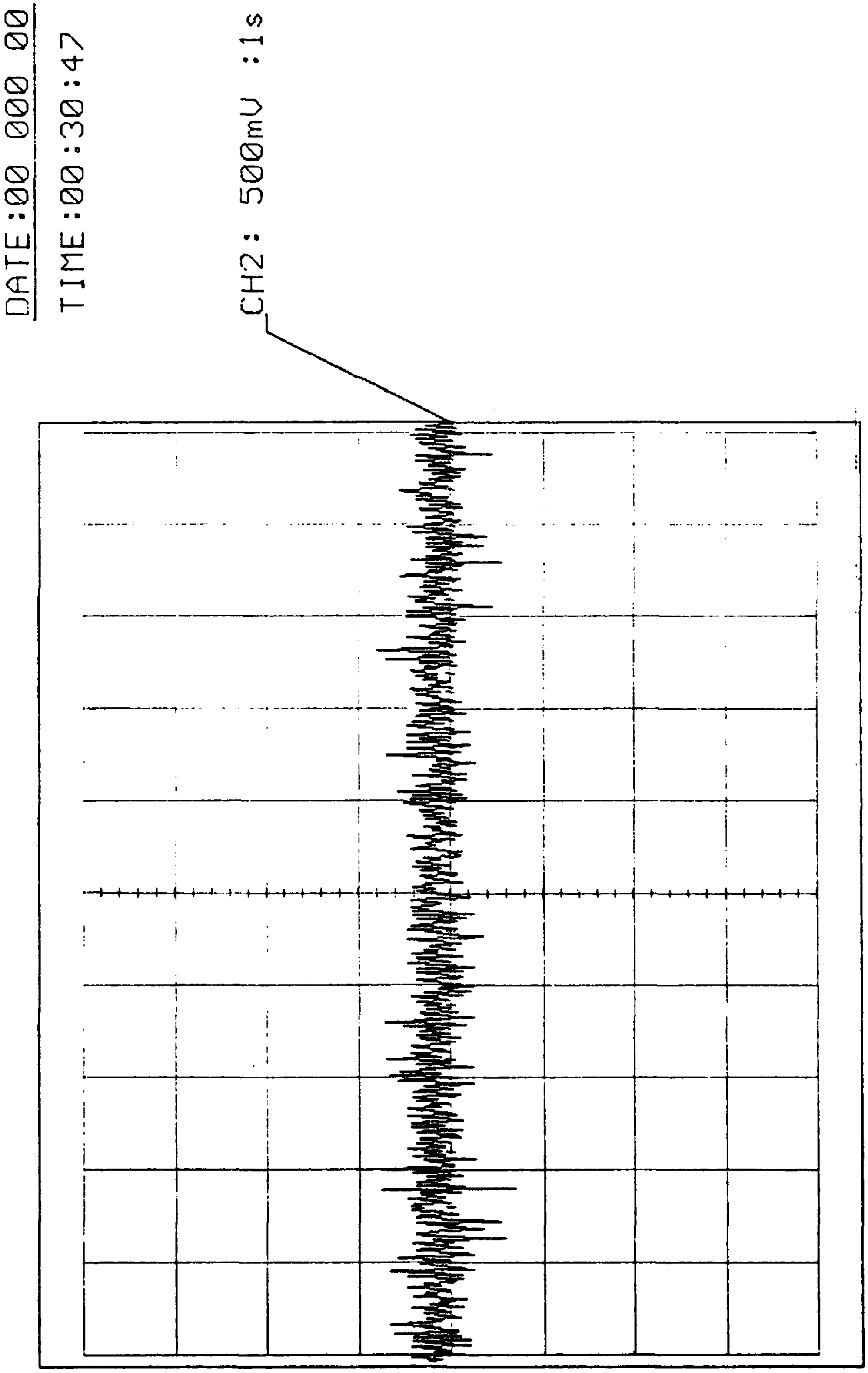
```
;******************************************************************************

                org     600h

Control         setb    3.6
                setb    3.5

                clr     weight_low          ;determine the weight (denominator)
                clr     weight_high
                mov     weight_low,reg1
                add     weight_low,reg2
                addb    weight_high,c       ;need to rl c 3-4?
                add     weight_low,reg3
                addb    weight_high,c
                add     weight_low,reg4
                addb    weight_high,c
                clr     control_high
                clr     control_low

_1st_reg        mov     reg,reg1
                mov     val,val1
                call    Multiply
                add     control_low,answer_lower
                add     control_high,answer_upper

_2nd_reg        mov     reg,reg2
                mov     val,val2
                call    Multiply
                add     control_low,answer_lower
                addb    control_high,c
                add     control_high,answer_upper

_3rd_reg        mov     reg,reg3
                mov     val,val3
                call    Multiply
                add     control_low,answer_lower
                addb    control_high,c
                add     control_high,answer_upper

_4th_reg        mov     reg,reg4
                mov     val,val4
                call    Multiply
                add     control_low,answer_lower
                addb    control_high,c
                add     control_high,answer_upper

                jmp     Divide

;******************************************************************************

Multiply        clr     answer_upper
                clr     answer_lower
                clr     counter
                cje     val,#00000000b,Return
                cje     reg,#00000000b,Return

Mult            clc
                rl      answer_lower
                rl      answer_upper
                jnb     val.7,Hi
                add     answer_lower,reg
Hi              jnb     val.7,Ho
                addb    answer_upper,c
Ho              clc
                rl      val
                inc     counter
                cjne    counter,#8,Mult

Return          ret
```

```
;*****************************************************************************

Divide          clr     counter1                ;clear counter1
                clr     counter2                ;clear counter2
                clr     answer

                mov     upper_divdiv2,weight_high
                mov     lower_divdiv2,weight_low ;move number __ into divdiv2
                clc                             ;clear carry
                rr      upper_divdiv2           ;rotate divdiv2 one bit to right (/2)
                rr      lower_divdiv2
                addb    lower_divdiv2,c         ;add c bit to dividiv2
                addb    upper_divdiv2,c

                clc                             ;clear carry
                call    count_zeros             ;call subroutine to count zeros

                jmp     long_div                ;jump to long_div

count_zeros     snb     weight_high.7           ;if 7th bit is "1", then return
                ret                             ;(return)
                clc
                rl      weight_low
                rl      weight_high             ;shift divisor one bit to left
                inc     counter1                ;and one to zeros counter
                jmp     count_zeros             ;check next bit

long_div        cja     weight_high,control_high,comp_counters
                cje     weight_high,control_high,upper_equal
subtract        sub     control_low,weight_low        ;subtract divisor from number
                sc
                dec     control_high
                sub     control_high,weight_high
                inc     answer                         ;add one to answer
comp_counters   cje     counter2,counter1,Remainder    ;if shifted to right as many zeros as
shifted to left, go to Output
                inc     counter2                ;add one to counter2
                clc                             ;clear carry register
                rl      answer                  ;shift answer one bit to left
                clc
                rr      weight_high             ;shift divisor  one bit to right
                rr      weight_low
                jmp     long_div                ;jump to long_div

upper_equal     cja     weight_low,control_low,comp_counters
                jmp     subtract

Remainder       cja     control_high,upper_divdiv2,Add
                cjb     control_high,upper_divdiv2,Outport
                cjb     control_low,lower_divdiv2,Outport   ;see if remainder can be rounded

Add             add     answer,#00000001b                   ;if so, then add one to answer

Outport         mov     RC,answer
                ljmp    Move_One
```

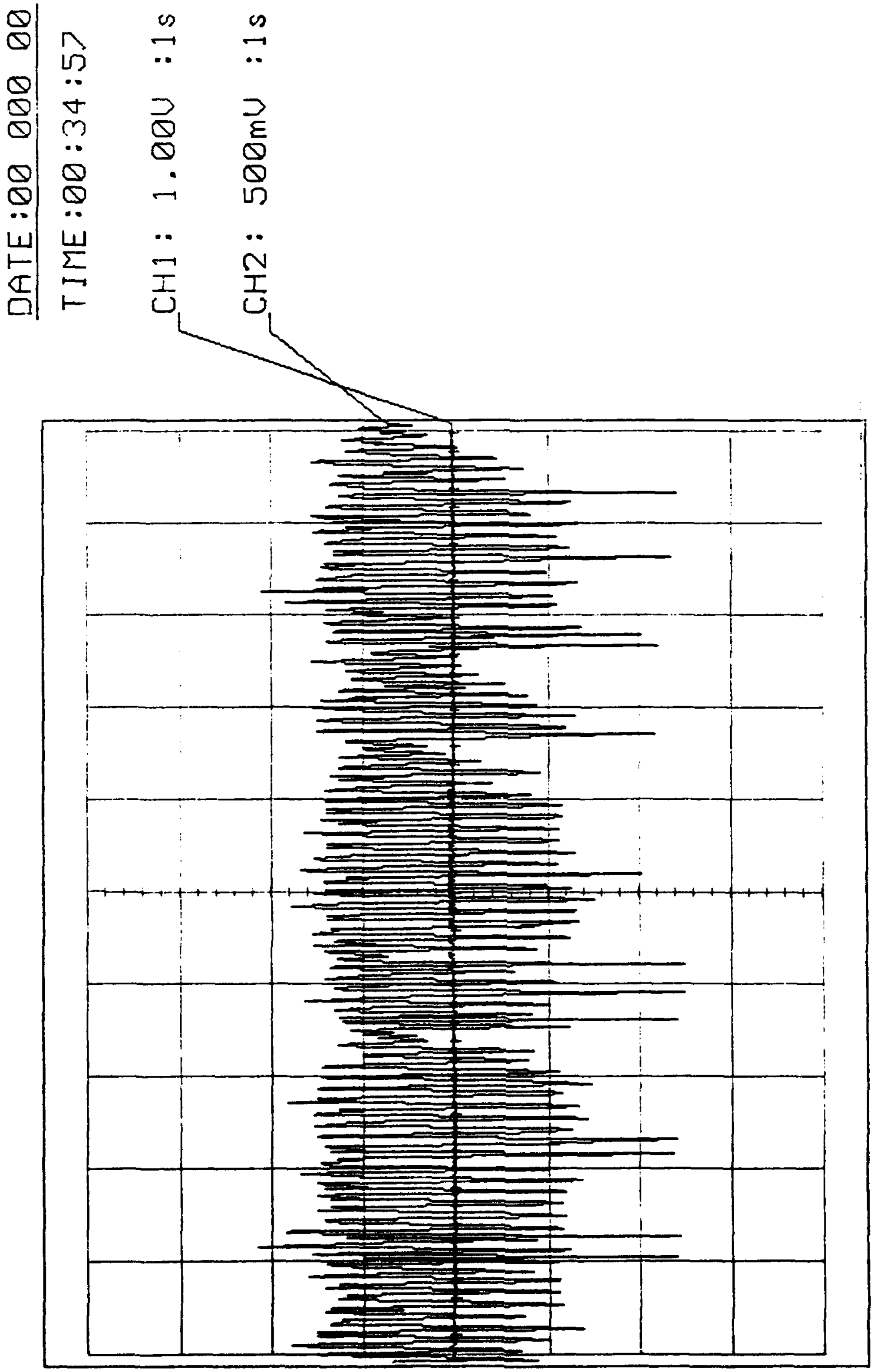
Appendix 9.6 - Program PWM

```
;31 Mar 94
;Program for pulse width modulation - Program pm.src

          DEVICE PIC16C54,HS_OSC,WDT_OFF,PROTECT_OFF

          RESET Set_Up

register  equ    08h
register1 equ    12h
counthi   equ    09h
countlo   equ    10h
count1    equ    11h

Set_Up    mov    !RB,#11111111b
          mov    !RA,#1100b
          clr    RA
          clr    RB

Start     mov    register,RB
          jb     register.7,Here;check sign
          clrb   RA.0
          comf   register,1
          inc    register
          jmp    Move

Here      setb   RA.0
          csne   register,#10000000b
          inc    register
Move      clrb   register.7
          mov    register1,register
          clc
          rr     register1
          addb   register1,c
          add    register,register1
          mov    counthi,register
          mov    countlo,#11111111b
          sub    countlo,register

          clrb   RA.1;"active low" PWM pulse
Pulse1    mov    count1,#04h
Pulse2    djnz   count1,Pulse2
          djnz   counthi,Pulse1

          setb   RA.1;"rest low" rest pulse
Rest1     mov    count1,#04h
Rest2     djnz   count1,Rest2
          djnz   countlo,Rest1

          jmp    Start
```

Appendix 9.71 - Plot of Motor Control Error Signal vs. Time (Tracking Platform noise.)



Appendix 9.72 - Plot of Motor Control Error Signal vs. Time (Tracking Platform centered on laser image.)



Appendix 9.73 - Plot of Motor Control Error Signal vs. Time (Tracking Platform tracking laser sweep in azimuth axis.)